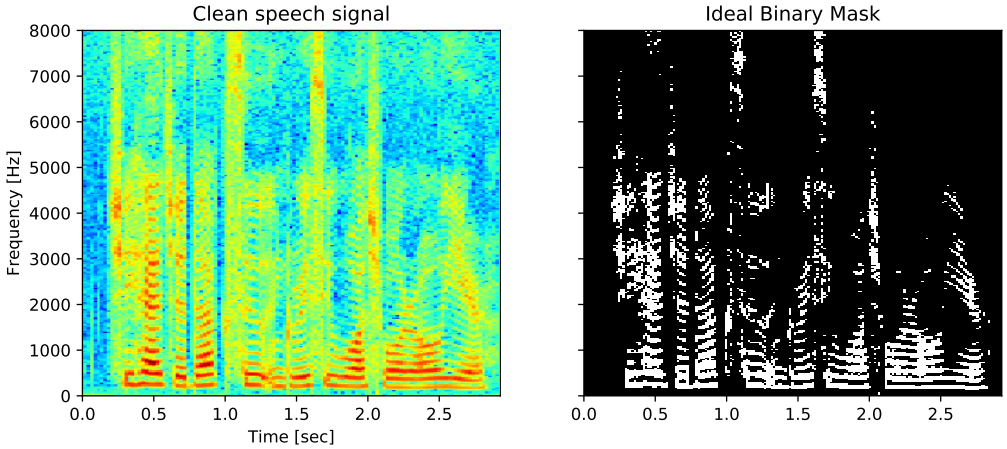
Own-Voice Retrieval for Hearing Assistive Devices

A Combined DNN-Beamforming Approach



Master's Thesis
Julius Garde
Mathematical Engineering
Aalborg University
2019

AALBORG UNIVERSITET

STUDENTERRAPPORT

Department of Electronic Systems
Department of Mathematical Sciences
Mathematical Engineering
Aalborg University
Fredrik Bajers Vej 7
9220 Aalborg
https://es.aau.dk
https://math.aau.dk

Title:
Own-Voice Retrieval for Hearing Assistive Devices: A Combined DNN-Beamforming Approach

Project Period:
September 2018 - May 2019

Project Group:
MATTEK10 Group 5.213 F

Participant(s):
Julius Garde

Supervisor(s):
Jesper Jensen
Zheng-Hua Tan
Wai-Yip Geoffrey Chan

Copies: 2

Page Numbers: 40

Date of Completion:
June 7st 2019

Abstract:

In this thesis, we propose a speech enhancement system for own-voice retrieval the presence of additive background. The system is designed with embedded devices in mind, specifically hearing assistive devices. The work focuses on discriminating noise-dominated time-frequency units from own-voice-dominated units, for which we employ a convolutional neural network to perform classification. Using only the classified time-frequency units, we show that a MVDR and a MWF beamformer can be constructed. Our results show that considerable improvements can be made in terms of perceived quality and intelligibility using the MVDR beamformer for selected noise types, whereas speech-shaped noise and babble-noise remains a challenge.

Preface

This Master's thesis is written by Julius Garde as part of the requirements for a Master of Science (MSc) degree in Engineering (Mathematical Engineering) at Aalborg University. The project was carried out between September 2018 and June 2019 with an extend of 60 ECTS-points. The IEEE-method is used in the report for referencing.

The scope of the project was formulated together with supervisors Jesper Jensen from Oticon A/S and Aalborg University, and Zheng-Hua Tan from Aalborg University, whom I would like to thank for their guidance, critique and helpful discussions during and leading up to the project. I would also like to thank Wai-Yip Geoffrey Chan from Queen's University, Kingston, who joined the team in October 2018. His insight on speech quality assessment and scientific methodology helped me during the project. Finally, I would like to thank Oticon A/S for providing the head-related transfer functions used in the experimental part of the project.

Regarding notation, all signals in this report are discrete in time. Capital letters will be used to denote variables in the frequency domain, while bold and underline are used to denote vectors and matrices, respectively. Furthermore, we will refer to the imaginary unit as it is done in engineering fields, thus j represents the imaginary unit. s for prerequisites, the reader is expected to be familiar with the Fourier transform of discrete time signals, temporal sampling of signals, stochastic processes and basic machine learning and optimization theory.

Aalborg University, June 7$^{\text{th}}$ 2019

Julius Garde
<jgarde14@student.aau.dk>

Contents

Introduction

In most if not all everyday situations, we encounter sounds which are either unwanted or uninteresting to us. These sounds are generally regarded as noises, which are interfering with the sounds that do have an immediate interest. An example hereof is a noisy car cabin. Here, the sounds generated by the car engine may be regarded as noise, whereas a GPS with voice navigation enabled may be the sound of interest. A crowded cafeteria is another example, where background chatter and clinking cutlery constitute the noise and an on-going conversation is the sound of interest. The noise can also originate from the desired sound itself, e.g. in highly reverberant environments, where the direct path of the sound of interest is concealed by late reflections of itself. In any case, the noise may cause issues for the listener if the listening conditions are particularly adverse.

Although the human auditory system has a remarkable capability to block out noises and focus the attention on the sounds of interest, it remains a challenging task in audio signal processing [2, 3]. The process of using signal processing tools for attenuating noise while preserving the desired signal (usually speech) is known as *noise reduction*, and has received a considerable amount of attention and research in the past several decades [4]. Noise reduction has applications in a variety of speech processing systems, including telecommunications [73], automatic speech recognition systems (ASR) [56] and hearing assistive devices [9]. These systems often operate in noisy environments, similar to the examples mentioned above, where noise-contaminated speech signals are recorded for e.g. transmission, processing or playback. For human listeners the purpose may be to improve the intelligibility or perceptual quality of the speech signal; for machine listeners the goal may be to improve ASR performance.

Noise reduction is generally a difficult task for various reasons. For once, the acoustic environment is often ever changing, which requires the design of a method that performs well in most situations. This usually requires a way of tracking the noise and/or the desired signal. Additionally, when designing a method for noise reduction, one is met with a trade-off between effective noise attenuation and introduced speech distortion [15]. This choice will be application specific; a speech recognition system may perform better when attenuating a competing speaker at the cost of minor speech distortion. On the other hand, speech distortion may be less tolerant

in systems made for human listening, e.g. a hearing aid.

A variety of methods exists for noise reduction. Linear filters are a common approach, where filter weights are specifically designed to achieve some goal, e.g. attenuate unwanted signal components. Linear filters are popular because of their simplicity, which makes them desirable from a computational point of view. Among linear filters is the minimum variance distortionless response (MVDR) beamformer, which is specifically designed to leave the desired signal undistorted. Another filter is the Wiener filter, which is derived from finding the optimal solution in the minimum mean squared error (MMSE).

Other approaches are based on computational auditory scene analysis (CASA) [77], which itself is based on the principles of the auditory scene analysis (ASA) [10]. CASA aims to develop speech separation algorithms which mimic the human auditory system. In [10], it is suggested that the human auditory system segregates mixed sound sources in two stages; segmentation and grouping. In the first stage, the acoustic mixture is decomposed into time-frequency segments. These segments are then grouped in the second stage using grouping cues such as periodicity, harmonicity and onset/offset synchrony [76]. One of the main computational goals of CASA is to estimate the ideal binary mask, which specifies which units in the time-frequency domain are predominantly speech or noise [46]. The estimated ideal binary mask can then be used directly as a gain function to attenuate noise-dominated tiles, or be used for further processing of the signal.

Scope of the Project

The objective of this thesis is to explore methods for own-voice retrieval in hearing assistive devices. Specifically, we utilize traditional beamforming techniques in combination with deep learning architectures to obtain an estimate of a hearing aid wearer's own voice. With a time-frequency mask as the computational goal, noise reduction can be cast as a supervised learning problem, for which deep neural networks are particularly suitable for due to their strong learning capabilities. In this thesis we explore methods of estimating a time-frequency mask for use in conjunction with traditional beamforming algorithms.

The purpose of the deep learning model is to extract spectral and learn the classification rule for discriminating between time-frequency bins with high and low SNRs. A sub-objective is to explore the possibilities of learning to specifically extract own-voice in the presence of competing speakers.

Chapter 1

System modeling

The purpose of this chapter is to develop a model for the output signal of the receiving microphone array in a hearing aid (HA). Using this model, the problem of estimating the desired own-voice speech signal from the noisy observations is formulated. Furthermore, assumptions regarding the model and their validity is discussed. At last, various performance measures are presented, which will be used for evaluation in the subsequent chapters.

1.0.1 Design and Evaluation Implications

Designing advanced signal processing algorithms for use in HAs poses several challenging tasks. For example, many HA devices are designed to be concealed behind or in the ear. As a result, HAs are often quite small in size, which means that the microphone configurations available are severely limited. Another concern is power consumption. Ideally, the power consumption should be low enough to keep the user from having to replace the batteries several times a day. The processing power is also limited. If the computational complexity becomes too high, a noticeable delay will be introduced in the system. In [66] a qualitative study was carried out to investigate the perception of delayed playback in HAs for own-voice signals. Test subjects started rating the experience as 'disturbing', when the processing delay exceeded 20-40 ms. Delays this long would certainly be unacceptable in any application where listening comfort is important. Even though a full implementation on a HA is beyond the scope of this thesis, these practical concerns should still be taken into consideration when assessing the performance of various de-noising algorithms in the following chapters.

1.1 Observation Model

In this thesis we consider the set-up depicted in Figure 1.1, where a HA wearer is equipped with multiple microphones and is situated in a noisy acoustic environment. When the HA wearer speaks, a noise-corrupted speech signal is received at the microphone arrays. The objective is then to recover the clean speech signal by suppressing the noise.

The noise environment may be composed of several noise sources, and each of these can either be diffuse noise (e.g. car interior) or localised noise impinging from a certain direction (e.g. a noisy fan). But since the individual noise sources are generally not of interest, they will be treated as a single entity.

The speech source is modelled as a *point source*, which is an infinitesimally small volume emitting spherical waves. These waves are characterised by having a constant sound pressure on spherical surfaces centered around the source, which makes them independent of direction [38, Sec 1.2]. Point sources approximate several real-world acoustic sources and will be a convenient model in the following section [65, Sec. 2.4.1].

The focus of this thesis is on *noise reduction* rather than *dereverberation*. For this reason, will consider the case of an (almost) free-field propagation environment with the HA wearer's head being the only obstacle.

1.1.1 Time-Domain Observation Model

Presume the HA wearer speaks and we measure a signal $y_m[n]$ at the m'th microphone at time index $n \in \mathbb{Z}$. The signal is assumed to consist of; (1) the clean speech signal $s[n]$ (convolved with the *mouth-to-mic* impulse response $h_m[n]$ between the speech source and the m'th microphone); and (2) an additive noise term $v_m[n]$, which accounts for the surrounding noise. The observation model is

$$\begin{aligned} y_m[n] &= (s[n] * h_m[n]) + v_m[n] \\ &= x_m[n] + v_m[n], \quad \text{for} \qquad m = 1, 2, \ldots, M, \end{aligned} \tag{1.1}$$

where $h_m[n]$ is the impulse response, $s[n]$ is the clean speech signal and $*$ denotes the usual convolution. The noise term $v_m[n]$ models both inherent noise in the system (e.g. thermal noise) and interference (such as competing speakers or background noise).

We make the following assumptions regarding the set-up:

1. The target speech signal $s[n]$ and the noise signal $v[n]$ are statistically uncorrelated.

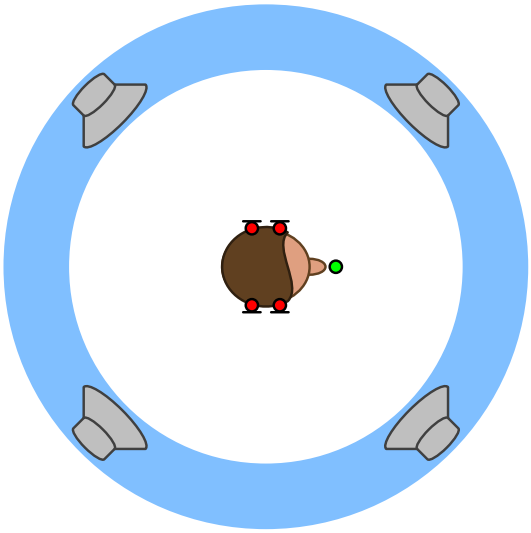


Figure 1.1: The acoustic scene. The HA wearer is equipped with four microphones (marked with red), and the target signal is the speech signal leaving the wearers mouth (marked with green). Except for the head, the signals are assumed to propagate in free field.

2. The signals in (1.1) are real and zero-mean.
3. The head-related impulse responses between the HA wearer and each of the microphones are known.

Regarding the validity of the above assumptions, we add the following comments. Assumption (1) is valid for many types of noise encountered in realistic set-ups, where the event producing the noise is unrelated to the person speaking. The assumption breaks, however, if e.g. the target speaker is fond of singing along to the car stereo. Another example is conversation, where the target speaker may be negatively correlated with the interfering speaker. Assumption (2) is valid since the signals are acoustic signals. Regarding Assumption (3), the head-related impulse response can be measured or estimated. And since the microphone array and target speech source are in fixed positions, the impulse response is not expected to change significantly as time progresses. The room impulse response, however, is dependent on an ever changing acoustic scene, thus it can not possibly be known in advance. Fortunately, late reflections of the target speech (those impinging 50-95 ms after the direct path) do not contribute to speech intelligibility, and can be considered as interference [12] [9, Sec. 1]. But the early reflections constitute a source of correlated noise, and these should be considered a part of the target speech signal.

Note that since the HRIR, $h[n]$, is assumed to be known, it suffices to obtain just one of the clean (but convolved) speech signals $x_m[n]$ in (1.1). Without loss of generality, we will choose microphone $m = 1$ as the *reference* microphone. Hence, $x_1[n]$

constitutes our desired signal. In practice, the microphone having the highest SNR can be selected as the reference microphone.

In the subsequent chapters, we will design de-noising filters which are based on the signal model in (1.1). Clearly, the performance of the filter is expected to improve by increasing the number of microphones in (1.1), since more measurements will be available for processing. However, this comes at the cost of increased requirements for the DSP chip, and possibly the need of a communication link between the two HAs to allow transmission of audio signals. For simplicity, we will therefore only consider the monaural case, corresponding to a single HA, equipped with two microphones. As we will see later, increasing the number of microphones in the algorithms is straight-forward, and thus is only a practical concern.

The signals in (1.1) are likely to be highly non-stationary. As a result, the signal statistics are going to be time dependent. This is an issue for obtaining accurate estimates of e.g. the covariance matrices of the signals, since these are often acquired by temporal averaging. The estimates may simply never converge. To overcome this, the signal can be segmented into smaller frames of samples in which the signal statistics are assumed to not change significantly. This property is called *quasi-stationarity*, and it turns out that this assumption is particularly valid for speech, where intervals of about 20 milliseconds ensure approximate *stationarity* [48].

Another approach is to solve the noise reduction problem in the STFT domain. Similarly to the time-domain approach, the signal is segmented into smaller frames using e.g. a sliding windowing function to appropriately handle the frame edges. The frames are then transformed into the frequency domain using a DFT, where the noise filtering is applied on the transform coefficients. Finally, the enhanced time-domain speech signal is synthesized from the estimated clean speech spectrum using the inverse DFT followed by an overlap-and-add method [6, Ch. 1]. This is the approach we will pursue in this thesis.

Besides being more computationally efficient than the time-domain approach [6, Ch. 1] [62], the STFT also acts as a decorrelator providing transform coefficients which are approximately uncorrelated in time and frequency [25, Ch. 2]. This allows processing the coefficients independently of each other. And although this assumption is often a gross simplification, it is nevertheless commonly used for convenience [25, Ch. 2]. Finally, while other transform domains can be used (e.g. Gabor wavelets [47], Karhunen-Loève expansions [4]), the STFT domain is particularly compelling because of the efficient FFT implementation [18], which makes the transform computationally cheap to use.

1.1.2 STFT-domain Observation Model

For a time-domain signal $y[n]$, we define the STFT as

$$Y(k,l) \triangleq \sum_{n=0}^{N-1} y[n+kD]w[n]\exp^{-2\pi ik\frac{1}{N}}, \tag{1.2}$$

where N denotes the DFT length, D is the filterbank decimation factor and $w[n]$ is a *windowing* function. For more on the STFT, see [20]. Assuming it exists, the STFT of (1.1) is

$$Y_m(k,l) = S(k,l)H_m(k) + V_m(k,l) \tag{1.3}$$
$$= X_m(k,l) + V_m(k,l), \tag{1.4}$$

where each term is the corresponding STFT-domain representation of the terms in (1.1), and k, l denote the frequency bin and time frame, respectively. Note that we drop the time frame indexing for H_m, since it is assumed to be time invariant. For convenience, we adopt the following vector notation

$$\boldsymbol{Y}(k,l) = S(k,l)\boldsymbol{H}(k) + \boldsymbol{V}(k,l) = \boldsymbol{X}(k,l) + \boldsymbol{V}(k,l), \tag{1.5}$$

where the terms are stacked in $M \times 1$-dimensional vectors across channels, that is, we define $\boldsymbol{Y}(k,l) := [Y_1(k,l), \ldots, Y_M(k,l)]^\top$ and similarly for $\boldsymbol{H}(k)$, $\boldsymbol{V}(k,l)$ and $\boldsymbol{X}(k,l)$. By further defining the vector

$$\boldsymbol{d}(k) := \left[1, \frac{H_2(k)}{H_1(k)}, \ldots, \frac{H_M(k)}{H_1(k)}\right]^\top = \frac{\boldsymbol{H}(k)}{H_1(k)}, \tag{1.6}$$

we get another formulation of the observation model

$$\boldsymbol{Y}(k,l) = X_1(k,l)\boldsymbol{d}(k) + \boldsymbol{V}(k,l), \tag{1.7}$$

provided that $H_1(k) \neq 0$. The elements of the vector $\boldsymbol{d} \in \mathbb{C}^M$ can be interpreted as the *relative transfer functions* between the reference microphone (where $m = 1$) and the remaining microphones [6, Sec. 4.1]. Using (1.7), the task of noise reduction is then to recover $X_1(k,l)$ given the noisy observations $\boldsymbol{Y}(k,l)$. Methods for doing exactly this are presented in Ch. 2.

1.2 Performance Measures

In order to compare and evaluate performance of the presented methods for own-voice retrieval, several performance measures are employed. The performance measures are divided into three categories: those related to (1) effective noise reduction, (2) speech quality and (3) speech intelligibility.

Segmental SNR

A widely used performance measure from the first category is the time-domain *segmental signal-to-noise ratio* (SEG-SNR). In this thesis, SEG-SNR will be used to measure the degree of noise reduction. For unprocessed and enhanced signals, $x[n]$ and $\hat{x}[n]$, is computed by averaging frame level SNR estimates as follows

$$\mathrm{SNR}_{\mathrm{SEG}} \triangleq \frac{10}{M} \sum_{m=0}^{M-1} \log_{10} \frac{\sum_{n=Nm}^{Nm+N-1} x^2[n]}{\sum_{n=Nm}^{Nm+N-1} (x[n] - \hat{x}[n])^2}, \tag{1.8}$$

where M is the number of frames in the signal, and N denotes frame length (typically chosen in the range of 15-20 milliseconds) [45, Sec. 3]. By framing the signal, one can choose to discard frames in the computation of the sum which contain very low speech energy (e.g. those during speech pauses), and thus have very large negative SNRs. By doing so, one avoids the bias in the measure associated with these frames. Likewise, frames above 35 dB do not reflect large perceptual differences. For these reasons, we choose to limit the values in the outer sum of (1.8) to the range of $[-10, 35]$ dB, as is commonly done [23]. Although widely used, SEG-SNR has been shown to correlate poorly with speech quality [45, Sec. 3.6].

PESQ

Another widely used performance measure for speech is *Perceptual Evalution of Speech Quality* (PESQ) [57]. PESQ is a test methodology for automated assessment of the speech quality. It compares a degraded signal to a reference signal, which in our case are either the unproccesed or processed noisy signal and the clean speech signal, respectively. The output of the PESQ algorithm is a value on a Mean Opinion Score (MOS) scale taking values between -0.5 (worst) and 4.5 (best). PESQ has been shown correlate greatly with speech quality and correlate moderately with speech intelligibility [45, Sec. 3.6]. For this reason, PESQ is chosen as the main indicator of speech quality[1]. For more information regarding the PESQ algorithm, see [57].

STOI and ESTOI

For speech intelligibility we employ two objective measures: *short-time objective intelligibility* (STOI) [67] and *extended* STOI (ESTOI) [33]. STOI measures the temporal correlation of short-time envelopes between the reference and degraded signals, and these correlations are then averaged across time and frequency bands to produce a single score in the range of -1 (worst) to 1 (best). STOI has been shown to be highly correlated with human speech intelligibility score [44, Sec. 11.4]. ESTOI is an extension of STOI and was proposed to address the finding that STOI performs poorly for certain signals [33, 72].

[1] Specifically the ITU-T standard `p.862` version of PESQ

Chapter 2

Spatial Filtering

Many speech processing systems operate in noisy environments, where a desired speech signal is received together with interfering signals. Examples of interfering signals are competing speakers (or other acoustic noise sources), electromagnetic noise and reverberation effects from the surroundings. These interfering signals are generally unwanted and may negatively impact the perceived speech quality and intelligibility. For systems involving automatic speech recognition, the accuracy may also suffer [56].

In single-microphone setups, various filtering methods exists for speech enhancement [5]. Usually, the objective involves estimating the desired speech signal while attenuating the unwanted signals. However, if the desired signal and the interfering signals occupy the same spectro-temporal frequency bands, then temporal filtering cannot separate the two signals [74].

In contrast to single-microphone filtering, an array of microphones can be employed to capture the impinging signals, leading to a multi-microphone setup. The array offers an additional dimension (space), which enables the use of *spatial filtering* methods. These methods can be used in conjunction with spectro-temporal filtering methods. The use of an array can be seen as capturing samples in *space*, and these samples are generally correlated. If the desired signals and the interference signals originate from different spatial locations, one can utilize the spatial information to separate the signals [37, p. 67-68].

In this chapter, we introduce two methods of utilizing a multi-microphone setup for speech enhancement: the *minimum variance distortionless response* (MVDR) beamformer and the *multi-channel Wiener filter* (MWF). These beamformers constitute the main body of the speech enhancement system proposed in this thesis. The chapter is structured as follows: first, the necessary theory behind spatial filtering is presented briefly. Then the MVDR and MWF are derived and compared. And finally, some details regarding the implementation of the beamformer are given.

2.1 Beamforming

A *beamformer* is formulated as a spatial filter, which uses the output of a sensor array to form a beam in a desired shape and direction. The beam can be considered as a directional pattern, which attenuates signals coming from other directions [37, 74]. In the previous chapter, we arrived at the following observation model of the noise-corrupted microphone signals in the STFT-domain

$$\begin{aligned} \boldsymbol{Y}(k,l) &= S_1(k,l)\boldsymbol{d}(k) + \boldsymbol{V}(k,l) \\ &= \boldsymbol{X}(k,l) + \boldsymbol{V}(k,l), \end{aligned} \tag{2.1}$$

where k and l denote the frequency bin and time frame, respectively. In the STFT domain, beamforming is carried out by computing the dot product between a complex-valued weight vector $\boldsymbol{W}(k,l) = [W_1(k,l), \ldots, W_M(k,l)]^{\top}$ and the array output

$$\begin{aligned} Z(k,l) &= \boldsymbol{W}^H(k,l)\boldsymbol{Y}(k,l) \\ &= \boldsymbol{W}^H(k,l)\left[S_1(k,l)\boldsymbol{d}(k) + \boldsymbol{V}(k,l))\right] \\ &= X_{fd}(k,l) + V_{fn}(k,l), \end{aligned} \tag{2.2}$$

where $X_{fd}(k,l) = \boldsymbol{W}^H(k,l)\boldsymbol{X}(k,l)$ is the filtered desired signal, and $V_{fn}(k,l) = \boldsymbol{W}^H(k,l)\boldsymbol{V}(k,l)$ is the filtered noise signal. After filtering, the time-domain audio signal can be recovered by applying the inverse STFT.

The specific choices of the weighting vector $\boldsymbol{W}$ depends on the chosen beamforming method. In this thesis, we consider the class of *statistically optimal* beamformers, which are derived from optimizing some objective function, e.g. minimize the output noise power. The solutions incorporate signal statistics of the observed data, which are generally unknown and time-varying, and therefore the solutions are only optimal in a statistical sense. For this reason, the performance is largely dependent on how well the signal statistics are estimated and subsequently tracked [37, p. 72].

Invoking the assumption of uncorrelatedness, the variance of the beamformer output (2.2) is

$$\begin{aligned} C_{ZZ}(k,l) &= \mathrm{E}\left[Z(k,l)Z^*(k,l)\right] \\ &= \boldsymbol{W}^H(k,l)\underline{\boldsymbol{C}}_{\boldsymbol{XX}}(k,l)\boldsymbol{W}(k,l) + \boldsymbol{W}^H(k,l)\underline{\boldsymbol{C}}_{\boldsymbol{VV}}(k,l)\boldsymbol{W}(k,l) \\ &= C_{X_{fd}X_{fd}}(k,l) + C_{V_{fn}V_{fn}}(k,l), \end{aligned}$$

where

$$\underline{\boldsymbol{C}}_{\boldsymbol{XX}}(k,l) \triangleq \mathrm{E}\left[\boldsymbol{X}(k,l)\boldsymbol{X}^H(k,l)\right] = S_1(k,l)\boldsymbol{d}(k)\boldsymbol{d}^H(k) \tag{2.3}$$

is the rank-one covariance matrix of the convolved speech signal, and

$$\begin{aligned} \underline{\boldsymbol{C}}_{\boldsymbol{VV}}(k,l) &\triangleq \mathrm{E}\left[\boldsymbol{V}(k,l)\boldsymbol{V}^H(k,l)\right], \\ C_{X_{fd}X_{fd}}(k,l) &\triangleq \mathrm{E}\left[X_{fd}(k,l)X_{fd}{}^*(k,l)\right], \\ \text{and} \quad C_{V_{fn}V_{fn}}(k,l) &\triangleq \mathrm{E}\left[V_{fn}(k,l)V_{fn}{}^*(k,l)\right] \end{aligned} \tag{2.4}$$

are the respective (co)-variances of $\boldsymbol{V}$, X_{fd} and V_{fn} [22, Sec. 9.2].

In the following sections, we will assume that the noise is not fully coherent across the microphones, which is a valid assumption in real-world setups due to sensor self-noise. As a result, $\underline{\boldsymbol{C}}_{\boldsymbol{VV}}(k,l)$ will be full-rank and thus its inverse exists [22, Sec. 9.2]. We now introduce the MVDR beamformer.

2.1.1 Minimum Variance Distortionless Response

The MVDR weights are obtained by minimizing the output power subject to a single linear constraint — that the array response has unity gain in the desired 'look direction' [21], i.e.

$$\boldsymbol{W}^H(k,l)\boldsymbol{d}(k) = 1, \tag{2.5}$$

where $\boldsymbol{d}(k)$ is the steering vector evaluated in the look direction. For the sake of readability, we will omit the time-frequency indices in the following. The power of the array response is

$$\begin{aligned} \mathrm{E}\left[\boldsymbol{Z}\boldsymbol{Z}^H\right] &= \mathrm{E}\left[\boldsymbol{W}^H\boldsymbol{Y}\boldsymbol{Y}^H\boldsymbol{W}\right] \\ &= \boldsymbol{W}^H\underline{\boldsymbol{C}}_{\boldsymbol{YY}}\boldsymbol{W}, \end{aligned} \tag{2.6}$$

and the full optimization problem is

$$\min_{\boldsymbol{W}} \boldsymbol{W}^H\underline{\boldsymbol{C}}_{\boldsymbol{YY}}\boldsymbol{W} \quad \text{subject to} \quad \boldsymbol{W}^H\boldsymbol{d} = 1. \tag{2.7}$$

The solution to (2.7) can be found by forming the complex Lagrangian

$$\mathcal{L}(\boldsymbol{W};\lambda) = \boldsymbol{W}^H\underline{\boldsymbol{C}}_{\boldsymbol{YY}}\boldsymbol{W} + \lambda(\boldsymbol{W}^H\boldsymbol{d} - 1) + \lambda^*(\boldsymbol{d}^H\boldsymbol{W} - 1),$$

where λ is the Lagrange multiplier [1, 16]. Then by taking the derivative with respect to $\boldsymbol{W}^*$, equating it zero and imposing the unity gain constraint

$$\frac{\partial}{\partial \boldsymbol{W}^*}\mathcal{L}(\boldsymbol{W};\lambda) = \underline{\boldsymbol{C}}_{XX}\boldsymbol{W} + \lambda\boldsymbol{d} \triangleq 0 \quad \text{s.t} \quad \boldsymbol{W}^H\boldsymbol{d} - 1 = 0, \tag{2.8}$$

we get the optimal weight in terms of the Lagrange multipliers

$$\boldsymbol{W}_{\mathrm{MVDR}} \triangleq -\lambda\underline{\boldsymbol{C}}_{\boldsymbol{YY}}^{-1}\boldsymbol{d}.$$

The solution to the optimization problem (2.7) is [16]

$$\boldsymbol{W}_{\mathrm{MVDR}} \triangleq \frac{\underline{\boldsymbol{C}}_{\boldsymbol{VV}}{}^{-1}\boldsymbol{d}}{\boldsymbol{d}^H\underline{\boldsymbol{C}}_{\boldsymbol{VV}}{}^{-1}\boldsymbol{d}}, \tag{2.9}$$

where

$$\boldsymbol{d}(k) := \left[1, \frac{H_2(k)}{H_1(k)}, \ldots, \frac{H_M(k)}{H_1(k)}\right]^\top = \frac{\boldsymbol{H}(k)}{H_1(k)} \tag{2.10}$$

is the relative transfer functions between the reference microphone ($m = 1$) and the remaining microphones.

Steering vector estimation

The MVDR solution (2.9) depends on the steering vector $\boldsymbol{d}(k)$, which in turn depend on the incident angle of the received signals. Thus the direction of arrival must be known to effectively suppress the output power in the other directions. In the case of a mismatch between the estimated DOA and the true DOA, the desired signal will end up being attenuated.

Instead, we choose to compute the steering vectors directly from the speech covariance matrix, $\underline{\boldsymbol{C}}_{\boldsymbol{XX}}$ as follows. By the assumption of a single directional speaker source in $\boldsymbol{X}(k,l)$, $\underline{\boldsymbol{C}}_{\boldsymbol{XX}}(k,l)$ is rank-one and hence has at most one non-zero eigenvalue. Hence it can be decomposed as

$$\underline{\boldsymbol{C}}_{\boldsymbol{XX}}(k,l) = \mathrm{E}\Big[|S(k,l)|^2 \boldsymbol{d}(k)\boldsymbol{d}^H(k)\Big]. \tag{2.11}$$

We now apply the eigenvalue decomposition to obtain

$$\underline{\boldsymbol{C}}_{\boldsymbol{XX}}(k,l) = \underline{\boldsymbol{Q}}\,\underline{\boldsymbol{\Lambda}}\,\underline{\boldsymbol{Q}}^{-1}, \tag{2.12}$$

where $\underline{\boldsymbol{Q}}$ contains the eigenvectors and $\underline{\boldsymbol{\Lambda}}$ is a diagonal matrix containing the eigenvalues. The eigenvector corresponding to the non-zero eigenvalue is then the vector $\boldsymbol{d}(k)$. Assuming in practice that $\underline{\boldsymbol{C}}_{\boldsymbol{XX}}$ is well-estimated, it would be close to a symmetric rank-one matrix. And in such case, the principal eigenvector has been shown to be a good estimate of the steering vector [78, 82]. In the following section we derive the multi-channel Wiener filter.

2.1.2 Multi-channel Wiener filter

Another method for multi-microphone noise reduction is the *multi-channel Wiener filter* (MWF). As we will see later, the MWF can be considered as an extension to the MVDR by post-filtering the beamformer output with a single-channel Wiener filter. For this reason, we expect the further filtering to only improve the noise suppression, at the cost of violating the constraint in the MVDR formulation.

The MWF, as proposed in [14], can be seen as an extension of the single-channel Wiener filter. Recall the output of the beamformer as the product between a weight vector and the array output

$$Z(k,l) = \boldsymbol{W}^H(k,l)\boldsymbol{Y}(k,l) \tag{2.13}$$

with

$$\boldsymbol{Y}(k,l) = \boldsymbol{X}(k,l) + \boldsymbol{V}(k,l). \tag{2.14}$$

Define the error signal $E(k,l)$ as the difference between the desired signal and the filter output, that is

$$E(k,l) \triangleq X_m(k,l) - Z(k,l). \tag{2.15}$$

By minimising the mean-squared error (MSE) of the error signal (2.15), the Wiener solution is obtained

$$\boldsymbol{W}_{\mathrm{MWF}}(k,l) = \arg\min_{\boldsymbol{W}} \mathrm{E}\left[|E(k,l)|^2\right]. \tag{2.16}$$

We now want to expand the MSE expression. By using

$$\begin{aligned} C_{YY}(k,l) &= \mathrm{E}\left[Y_m(k,l)Y_m{}^*(k,l)\right], && \text{power of } Y_m \\ \boldsymbol{C}_{\boldsymbol{Y}Z}(k,l) &= \mathrm{E}\left[\boldsymbol{Y}(k,l)Z^*(k,l)\right], && \text{cross-covariance vector of } \boldsymbol{Y} \text{and } Z \\ \underline{\boldsymbol{C}}_{\boldsymbol{YY}}(k,l) &= \mathrm{E}\left[\boldsymbol{Y}(k,l)\boldsymbol{Y}^H(k,l)\right], && \text{auto-covariance matrix of } \boldsymbol{Y} \end{aligned}$$

the MSE can be rewritten as

$$\begin{aligned} \mathrm{E}\left[|X_m - Z|^2\right] &= \mathrm{E}\left[(X_m - \boldsymbol{W}^H\boldsymbol{Y})(X_m{}^* - \boldsymbol{Y}^H\boldsymbol{W})\right] \\ &= C_{XX} - \boldsymbol{W}^H\boldsymbol{C}_{\boldsymbol{Y}X} - \boldsymbol{C}_{\boldsymbol{Y}X}{}^H\boldsymbol{W} + \boldsymbol{W}^H\underline{\boldsymbol{C}}_{\boldsymbol{YY}}\boldsymbol{W}, \end{aligned} \tag{2.17}$$

where the indices k and l have been omitted. Note that the MSE (2.17) is real-valued and non-negative for all k. By assuming uncorrelatedness across frequency bins, minimising the sum of errors for all k corresponds to minimising the error for each frequency bin individually [9, Sec. 3.2]. Thus the frequency bins can be treated individually. Furthermore, the left-hand side is a quadratic function of $\boldsymbol{W}$. This means that the optimal solution can be obtained by setting the gradient of (2.17) with respect to $\boldsymbol{W}$ equal to zero, that is

$$\frac{\partial}{\partial \boldsymbol{W}}\mathrm{E}\left[|X_m - Z|^2\right] = -2\boldsymbol{C}_{\boldsymbol{Y}X} + 2\underline{\boldsymbol{C}}_{\boldsymbol{YY}}\boldsymbol{W} \triangleq \boldsymbol{0}, \tag{2.18}$$

where $\boldsymbol{0}$ is the $M \times 1$ null vector. The solution is found from the multi-channel Wiener-Hopf equations [9, Sec. 3.2]

$$\begin{aligned} &\underline{\boldsymbol{C}}_{\boldsymbol{YY}}(k,l)\boldsymbol{W}_{\mathrm{MWF}}(k,l) = \boldsymbol{C}_{\boldsymbol{Y}X}(k,l) \\ &\quad\Longrightarrow\ \boldsymbol{W}_{\mathrm{MWF}}(k,l) = \underline{\boldsymbol{C}}_{\boldsymbol{YY}}{}^{-1}(k,l)\boldsymbol{C}_{\boldsymbol{Y}X}(k,l). \end{aligned} \tag{2.19}$$

It should be emphasised that the above solution does not depend on the array configuration or DOA of the signals. It does, however, require knowledge of the desired signal $X_m(k,l)$ (through $\boldsymbol{C}_{\boldsymbol{Y}X}(k,l)$), which is unknown. The filter is therefore not realisable in its current form. Instead we invoke the assumption of the desired signal and noise signal being uncorrelated, which enables us to write the solution as

$$\boldsymbol{W}_{\mathrm{MWF}}(k,l) = \underline{\boldsymbol{C}}_{\boldsymbol{YY}}{}^{-1}(k,l)\left[\underline{\boldsymbol{C}}_{\boldsymbol{YY}}(k,l) - \underline{\boldsymbol{C}}_{\boldsymbol{VV}}(k,l)\right]\boldsymbol{I}, \tag{2.20}$$

where $\boldsymbol{i}$ is an M-dimensional unit vector with a '1' at the m'th entry corresponding to the reference channel. A method for obtaining $\underline{\boldsymbol{C}}_{\boldsymbol{YY}}(k,l)$ and $\underline{\boldsymbol{C}}_{\boldsymbol{VV}}(k,l)$ is presented in Sec. 2.2. In the following section, we compare the MWF with the MVDR beamformer for speech enhancement and show a resemblance between the two.

2.1.3 Comparing the MVDR and the MWF

The MWF can be factored into a MVDR beamformer and a single-channel WF by using the Sherman-Morrison[1] matrix identity. Recall from the observation model that

$$\boldsymbol{Y}(k,l) = \boldsymbol{X}(k,l) + \boldsymbol{V}(k,l) = X_1(k,l)\boldsymbol{d} + \boldsymbol{V}(k,l). \tag{2.21}$$

For ease of notation, we will omit the frequency bin and time frame indices, k and l, in the following derivation. By the assumption of desired speech and noise being uncorrelated, we get

$$\boldsymbol{C}_{\boldsymbol{Y}X_1} = C_{X_1X_1}\boldsymbol{d} \tag{2.22}$$

$$\text{and}\, \underline{\boldsymbol{C}}_{\boldsymbol{YY}} = C_{X_1X_1}\boldsymbol{d}\boldsymbol{d}^H + \underline{\boldsymbol{C}}_{\boldsymbol{VV}}. \tag{2.23}$$

Using the above, the MWF solution may now be written

$$\boldsymbol{W}_{\text{MWF}} = \underline{\boldsymbol{C}}_{\boldsymbol{YY}}{}^{-1}\underline{\boldsymbol{C}}_{\boldsymbol{XX}}, = \left[C_{X_1X_1}\boldsymbol{d}\boldsymbol{d}^H + \underline{\boldsymbol{C}}_{\boldsymbol{VV}}\right]^{-1} C_{X_1X_1}\boldsymbol{d}. \tag{2.24}$$

For a square matrix $\underline{\boldsymbol{A}} \in \mathbb{R}^{M\times M}$ and column vectors $u, v \in \mathbb{R}^M$, the Sherman-Morrison matrix identity [52, Sec. 3.2] states that

$$\left[\underline{\boldsymbol{A}} + \boldsymbol{u}\boldsymbol{v}^T\right]^{-1} = \underline{\boldsymbol{A}}^{-1} - \frac{\underline{\boldsymbol{A}}^{-1}\boldsymbol{u}\boldsymbol{v}^T\underline{\boldsymbol{A}}^{-1}}{1 + \boldsymbol{v}^T\underline{\boldsymbol{A}}^{-1}\boldsymbol{u}}, \tag{2.25}$$

provided that $\underline{\boldsymbol{A}}$ and $\underline{\boldsymbol{A}} + \boldsymbol{u}\boldsymbol{v}^T$ are invertible. By using this result with the following substitutions

$$\underline{\boldsymbol{A}} = \underline{\boldsymbol{C}}_{\boldsymbol{VV}}, \quad \boldsymbol{u} = \boldsymbol{v} = \sqrt{C_{X_1X_1}}\boldsymbol{d}, \tag{2.26}$$

we can write (2.24) as

$$\boldsymbol{W}_{\text{MWF}} = \left[\underline{\boldsymbol{C}}_{\boldsymbol{VV}}{}^{-1} - \frac{C_{X_1X_1}\underline{\boldsymbol{C}}_{\boldsymbol{VV}}{}^{-1}\boldsymbol{d}\boldsymbol{d}^H\underline{\boldsymbol{C}}_{\boldsymbol{VV}}{}^{-1}}{1 + C_{X_1X_1}\boldsymbol{d}^H\underline{\boldsymbol{C}}_{\boldsymbol{VV}}{}^{-1}\boldsymbol{d}}\right] C_{X_1X_1}\boldsymbol{d} \tag{2.27a}$$

$$= \left[1 - \frac{C_{X_1X_1}\boldsymbol{d}^H\underline{\boldsymbol{C}}_{\boldsymbol{VV}}{}^{-1}\boldsymbol{d}}{1 + C_{X_1X_1}\boldsymbol{d}^H\underline{\boldsymbol{C}}_{\boldsymbol{VV}}{}^{-1}\boldsymbol{d}}\right] C_{X_1X_1}\underline{\boldsymbol{C}}_{\boldsymbol{VV}}{}^{-1}\boldsymbol{d} \tag{2.27b}$$

$$= \left[\frac{C_{X_1X_1}}{1 + C_{X_1X_1}\boldsymbol{d}^H\underline{\boldsymbol{C}}_{\boldsymbol{VV}}{}^{-1}\boldsymbol{d}}\right] \underline{\boldsymbol{C}}_{\boldsymbol{VV}}{}^{-1}\boldsymbol{d} \tag{2.27c}$$

$$= \underbrace{\frac{\underline{\boldsymbol{C}}_{\boldsymbol{VV}}{}^{-1}\boldsymbol{d}}{\boldsymbol{d}^H\underline{\boldsymbol{C}}_{\boldsymbol{VV}}{}^{-1}\boldsymbol{d}}}_{\text{MVDR}} \underbrace{\frac{C_{X_1X_1}}{C_{X_1X_1} + (\boldsymbol{d}^H\underline{\boldsymbol{C}}_{\boldsymbol{VV}}{}^{-1}\boldsymbol{d})^{-1}}}_{\text{Single-channel WF}}, \tag{2.27d}$$

where the matrix identity (2.25) is applied in Eq. (2.27a), and $\underline{\boldsymbol{C}}_{\boldsymbol{VV}}{}^{-1}$ is moved outside the brackets in Eq. (2.27b). In Eq. (2.27c), we use that $\boldsymbol{d}^H\underline{\boldsymbol{C}}_{\boldsymbol{VV}}{}^{-1}\boldsymbol{d}$ is a real-valued scalar. Thus the optimal filter can be factorized into a MVDR part and a single-channel MWF.

[1] This is a special case of the Woodbury matrix identity.

2.2 Estimating the Necessary Signal Statistics

In the preceding sections, we arrived at optimal solutions to the noise reduction problem which depend on estimates of the speech- and noise covariances. Since the speech and noise signals are not known in isolated forms, obtaining accurate estimates of these covariances can be difficult. Furthermore, we are generally limited to just a single realization of the noisy process.

In order to estimate the covariance matrix of e.g. the noise, we need a method of obtaining samples hereof. In the observation model (1.7), noise is always assumed present. Speech, on the other hand, contains natural pauses in which only the noise term $\boldsymbol{V}(k,l)$ is observed, i.e. $\boldsymbol{Y}(k,l) = \boldsymbol{V}(k,l)$. Hence by identifying the time-frequency units of noise-only, samples of $\boldsymbol{V}(k,l)$ can be obtained in isolation. Using these samples, $\underline{\boldsymbol{C}}_{\boldsymbol{VV}}(k,l)$ can be estimated by e.g. the sample covariance [63]

$$\hat{\underline{\boldsymbol{C}}}_{\boldsymbol{VV}}(k,l) = \frac{1}{\mathrm{Card}(\mathbb{L})} \sum_{\ell \in \mathbb{L}} \boldsymbol{Y}(k,\ell)\boldsymbol{Y}^H(k,\ell), \tag{2.28}$$

where $\ell \in \mathbb{L}$ denotes the time frames containing noise-only and $\mathrm{Card}(\mathbb{L})$ is the cardinality of this set.

The speech-and-noise and noise-only time-frequency units can conveniently be specified using a *mask* defined as follows. Let $X_m(k,l)$ and $V_m(k,l)$ be defined as in (1.4), where $k = 0, 1, \ldots, K$ denotes frequency bin, $l = 0, 1, \ldots, L$ denotes time frame and $m \in \{1, 2\}$ denotes channel. The *ideal binary mask* (IBM) for the m'th channel is defined as the $K \times L$ matrix whose entries are

$$\mathrm{IBM}(k,l) = \begin{cases} 1 & \text{if } |X_m(k,l)| > \tau |V_m(k,l)| \\ 0 & \text{else} \end{cases}, \tag{2.29}$$

where $\tau \in \mathbb{R}_+$ is a threshold value. The threshold τ controls what constitutes a speech-and-noise unit, for example, stetting τ high means only high SNR time-frequency units are discarded for the purpose of estimating $\hat{\underline{\boldsymbol{C}}}_{\boldsymbol{VV}}(k,l)$. In computational auditory scene analysis (CASA), a threshold corresponding to 0 dB (i.e. $\tau = 1$) is commonly used, however slightly higher or lower thresholds may also be beneficial [42]. It should also be noted that the mask requires explicit knowledge of $X_m(k,l)$ and $V_m(k,l)$, and must therefore be estimated. In Figure 2.1, a noisy speech signal and an ideal binary mask is shown for $\tau = 1$.

As the theoretical covariance matrix may change over time, the sample covariance (2.28) relies on continuously being replenished with new samples to accurately track changes. In order to do so, the estimate can be initialized using a handful of samples and then be updated using an online estimation scheme as more observations become available. One way of doing this is by using recursive smoothing

$$\hat{\underline{\boldsymbol{C}}}_{\boldsymbol{VV}}(k,\ell+1) = \nu_{\boldsymbol{V}} \hat{\underline{\boldsymbol{C}}}_{\boldsymbol{VV}}(k,\ell) + (1-\nu_{\boldsymbol{V}}) \boldsymbol{Y}(k,\ell)\boldsymbol{Y}^H(k,\ell), \tag{2.30}$$

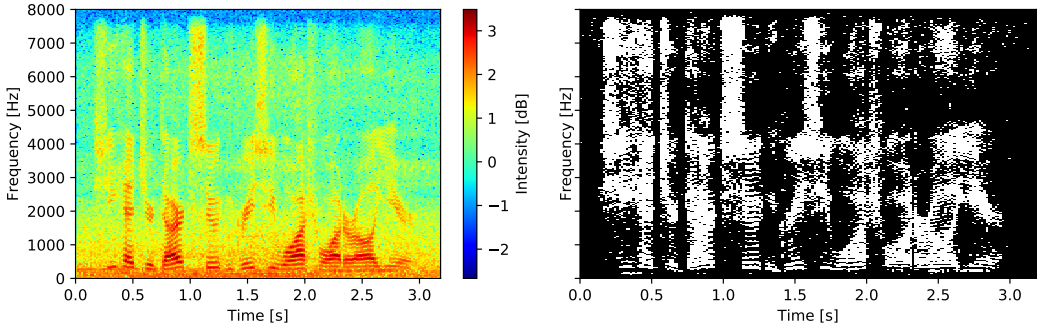


Figure 2.1: Example of a noisy speech signal (left) and the corresponding ideal binary mask (right).

where ℓ and $\ell + 1$ denote consecutive noise-only units and $\nu_{\boldsymbol{V}} \in [0, 1]$ is the *forgetting factor* associated with $\boldsymbol{V}$ [55, Sec. 1.3]. In a similar manner, $\hat{\underline{\boldsymbol{C}}}_{\boldsymbol{YY}}(k, l)$ can be estimated except $\boldsymbol{Y}(k, l)$ is always observed. Finally, a covariance matrix estimate of $\boldsymbol{X}(k, l)$ is obtained as $\hat{\underline{\boldsymbol{C}}}_{\boldsymbol{XX}}(k, l) = \hat{\underline{\boldsymbol{C}}}_{\boldsymbol{YY}}(k, l) - \hat{\underline{\boldsymbol{C}}}_{\boldsymbol{VV}}(k, l)$. Though, it should be noted that during speech-and-noise, $\hat{\underline{\boldsymbol{C}}}_{\boldsymbol{VV}}(k, l)$ is not updated even though $\boldsymbol{V}(k, l)$ is observed. This is a problem for noise with rapidly changing statistics, since $\hat{\underline{\boldsymbol{C}}}_{\boldsymbol{YY}}(k, l)$ may at times be estimated using past samples (say 100 ms ago). Because of this, it is commonly assumed that the noise statistics are not changing too rapidly, such that an estimate from the recent past still valid at the current time [25, Ch. 2].

The forgetting factor in (2.28) controls how fast the covariance matrix adapts to changes and should be chosen carefully. On the one hand, it cannot be too small; otherwise the estimate will then largely depend on the new samples, which may make the estimate fluctuate a lot and possibly degrade the performance of the beamformer. On the other hand, a very large factor makes the estimate incapable of tracking short-term changes, which may also degrade performance [4, Sec. 7.4.2]. Different noise types may also have different optimal parameter choices: for example, a stationary noise type may require a very slow decay since the signal statistics are constant, whereas highly non-stationary noise will require fast adaptation.

Experiments were carried out in order to select the forgetting factors, for which $\nu_y = \nu_v = 0.99$ gave decent results in terms of the four objective performance measures (see Sec. 1.2). This corresponds to averaging the covariance matrices over $\frac{1}{1-\nu} = 100$ time frames, corresponding to 1.616 seconds. Inspired by [24, 43], we choose a DFT size of of 512 samples for the STFT (corresponding to 32 ms of audio) with 50% overlap between consecutive frames. On the one hand, a sufficiently high frequency resolution is necessary to separate desired speech and noise. On the other hand, a frequency resolution too high may degrade the reliability of the spatial covariance matrix estimates, since it may take a long time before sufficiently many observations of speech are observed [24]. Furthermore, the signal is downsampled

to 16 kHz and a square-root Hann window is applied prior to transformation. In Fig. 2.2, a noisy speech signal is enhanced using a MVDR and MWF beamformer.

Although better performance may be achieved using different pairs of forgetting factors for each type of noise, it is beyond the scope of this thesis. Instead we choose to determine a one-fits-all pair across multiple noise types and keep it fixed for the remaining of the thesis. The results obtained using these forgetting factors will therefore also serve as the theoretical upper limit of the full own-voice retrieval system (assuming a perfectly estimated binary mask).

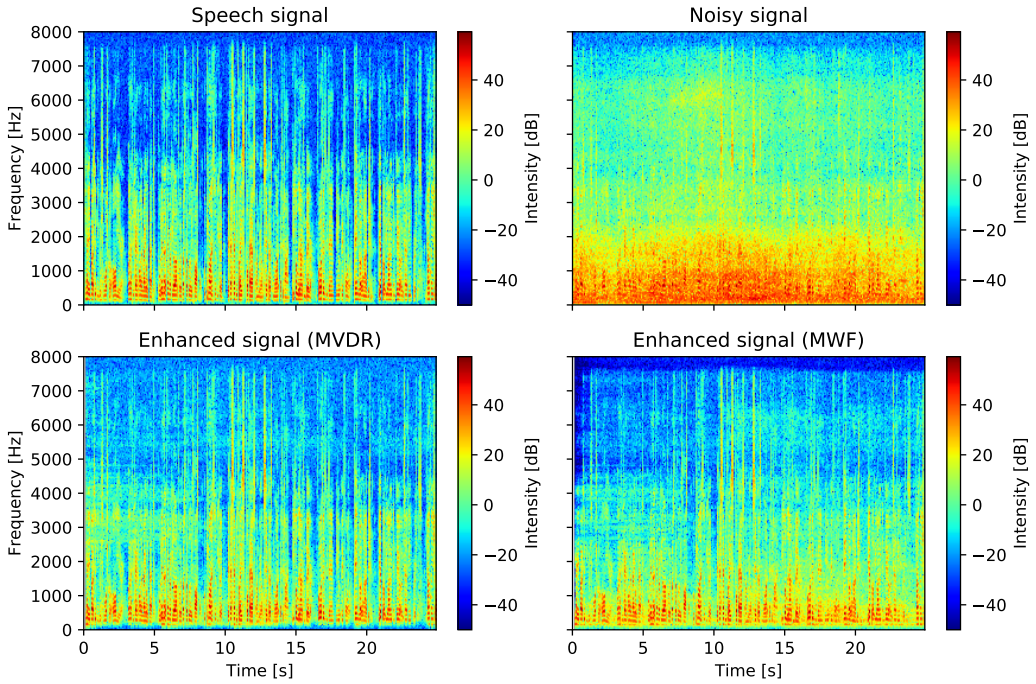


Figure 2.2: Noisy speech signal enhancement using the MVDR (bottom left) and MWF (bottom right) beamformer with $\nu_y = \nu_v = 0.99$. The signals are synthesized according to App. A.1.

Chapter 3

Deep Neural Networks

The beamformers derived in the previous chapter depend on accurate signal statistics estimates, and in order to obtain these, a machine learning model is employed. The purpose of the model is not to estimate these statistics directly, but rather to learn the complicated mapping between noisy time-frequency representations and the corresponding ideal binary masks (see Eq. (2.29)). The entries of the estimated mask can be interpreted as the posterior probability of a given time-frequency tile being speech-dominated. Using this information, the desired covariance matrices required for beamforming can be computed.

The goal of this chapter is to design a deep neural network capable of estimating the desired ideal binary mask, which will be used as part of the full system in the proceeding chapters. Before presenting the proposed model, we briefly summarize relevant theory regarding deep neural networks and training hereof. For a more in-depth cover of deep learning and practical methodology, the reader is referred to [19].

The chapter is structured as follows: In Sec. 3.1, a brief introduction to deep neural networks are given. In Sec. 3.2 we motive and discuss the design choices of the proposed model. Lastly, we carry out experiments for hyperparameter optimization and model selection.

3.1 Deep Neural Networks

When designing machine learning models, the goal is usually to approximate some unknown function f^*. For example, in a classification problem, one seeks a function f^*, which maps an input vector $\boldsymbol{x} \in \mathbb{R}^d$ to a category $\boldsymbol{y} \in \mathbb{R}^K$. Typically, the unknown function is approximated using a parameterized model $\boldsymbol{y} = f(\boldsymbol{x}; \boldsymbol{\theta})$, where $\boldsymbol{\theta}$ is a vector of adjustable parameters belonging to some vector space $\boldsymbol{\Theta}$. The parameters $\boldsymbol{\theta}$ are then *learned* by the model through an iterative method using a pre-defined set of input vectors $\boldsymbol{x}$ accompanied by the corresponding desired categories $\boldsymbol{y}$. This is typically accomplished using gradient descent via the *back-propagation* algorithm

[8][60]. The procedure of inferring a mapping using labelled data is called *supervised learning* [19, Ch. 6].

The type of parametric models considered in this thesis are called *deep feedforward networks*. These are composed of a chain of functions $f^{(j)}$, called *layers*, which themselves may consist of multiple functions. For a network consisting of J layers, we have

$$f(\boldsymbol{x};\boldsymbol{\theta}) = f^{(J)}(f^{(J-1)}(\dots(f^{(1)}(\boldsymbol{x}))), \quad 1 \leq j \leq J, \tag{3.1}$$

where $f^{(j)} : \mathbb{R}^{K_{j-1}} \to \mathbb{R}^{K_j}$ and $K_1, K_2, \dots, K_{J-1}$ are the dimensions of the so-called *hidden layers* [8]. A common layer consists of an affine linear transformation followed by a non-linear transformation (which operates component-wise)

$$f^{(j)}(\boldsymbol{x}^{(j-1)}) = h^{(j)}\left(\underline{\boldsymbol{W}}^{(j)\top}\boldsymbol{x}^{(j-1)} + \boldsymbol{b}^{(j)}\right), \tag{3.2}$$

where $\underline{\boldsymbol{W}}^{(j)} \in \mathbb{R}^{K_{j-1}\times K_J}$ and $\boldsymbol{b}^{(j)} \in \mathbb{R}^{K_j}$ are the parameters, $h^{(j)} : \mathbb{R}^{K_j} \to \mathbb{R}^{K_j}$ denotes a non-linear transformation (called an *activation function*), $\boldsymbol{x}^{(j-1)}$ is the output of the previous layer and superscript (j) denotes affiliation with layer j. The role of the activation function is to introduce non-linearity in the model, which enables learning of more complex functions. Common choices are sigmoidal functions or variants of the *rectified linear unit* (ReLU) [7, Ch. 5][8].

Feedforward refers to the uni-directional flow of information; $\boldsymbol{x}$ is propagated through the network from the first layer to the last. The number of layers J is referred to as the *depth* of the model, and models with more than a few layers are called *deep learning* models. The mapping (3.1) can conveniently be represented as a weighted acyclic directed graph (see Fig. 3.1). Specifically, instead of thinking of a layer as representing a single vector-to-vector mapping, it can be thought of as consisting of many nodes (also called *neurons*), each representing a vector-to-scalar function, that acts in parallel [19, Ch. 5]. Each node is interconnected only with nodes in the adjacent layers. For example, the nodes in (3.2) each compute

$$x_k^{(j)} = h^{(j)}\left(\sum_i^{K_{j-1}} w_{ki}^{(j)} x_i^{(j-1)} + b_k^{(j)}\right), \quad k = 0, 1, \dots, K_j, \tag{3.3}$$

where $i = 0, 1, \dots, K_{j-1}$ is the number of in-going connections to the k'th node in layer j.

3.1.1 Learning From Data

When fitting the model to data, we seek to approximate f^* by essentially providing the model with function values of f^* evaluated at discrete training points. That is, for each vector $\boldsymbol{x}_n$ in a *training dataset* of N feature-label pairs, $\mathbb{X} = \{(\boldsymbol{x}_n, \boldsymbol{y}_n)\}_{n=1}^N$, the model is optimized to match the corresponding label $\boldsymbol{y}_n \approx f(\boldsymbol{x}_n;\boldsymbol{\theta})$ by tuning the

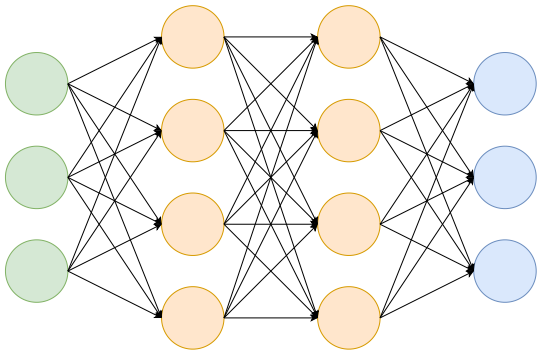


Figure 3.1: Feedforward neural network represented as an acyclic directed graph with nodes arranged in hierarchical layers. (green) input layer, (yellow) hidden layers, (blue) output layer.

parameters $\boldsymbol{\theta}$ [19, Ch. 6]. A loss function is used to measure the disparity between the network predictions and the labels, which typically is used to guide a gradient-based optimization algorithm. For binary classification tasks, a popular loss function is the binary *cross-entropy* [7, Ch. 4] defined as follows

$$J(\boldsymbol{\theta}) = -\frac{1}{N}\sum_{n=1}^{N} \boldsymbol{y}_n \log \boldsymbol{x}_n^{(J)} + (1-\boldsymbol{y}_n)\log(1-\boldsymbol{x}_n^{(J)}). \tag{3.4}$$

where $\boldsymbol{x}_n^{(J)}$ is the network prediction and $\boldsymbol{y}_n$ is a vector encoding the true label. The *training loss* is defined as the average of the losses (3.4) computed over the entirety of the training dataset. Although being the primary measure of how well the model fits the data, the training loss is a bad indicator of true model performance, since it can be made arbitrarily small by allowing the model to essentially "memorize" the dataset. This is the extreme case of the concept known as *overfitting*. Instead, the learned model should be evaluated based on performance on unseen data, since this resembles how the model performs when deployed in the real world. For this reason, an additional *testing* dataset can be introduced for the sole purpose of evaluating the learned model. Here, it is particularly important that the disjoint test set is not used in any way to infer the hyperparameters. Alternatively, a third *validation* dataset can be introduced for the purpose of choosing hyperparameters or deciding between trained models. This is to ensure that the testing set is disjoint from the training set to better represent an accurate estimate of performance on unseen data [19, Ch. 5]. In the remaining chapters, special attention will be made paid to ensure overfitting does not occur.

Deep learning models belong to a class of methods called *representative learning* methods. These methods are able to learn the discriminative representations required for e.g. classification from raw data. In contrast to conventional machine learning models, which often require hand-crafted features, much of the feature extraction stage is incorporated into the model, where the relevant transforms are

learned by the layers. As the number of layers increase, so does the abstraction level of the later layers, which more selectively can amplify the discriminative aspects while suppressing irrelevant information [41]. Although circumventing the need for carefully designed features is convenient, domain knowledge can still be applied to potentially accelerate learning by using relevant pre-processing transforms on the input. For example, DNNs operating on time-frequency domain features have been shown to outperform similar networks operating on time-domain features [26][75].

Neural networks have remarkable modelling capabilities. In fact, several theorems exist stating that even two-layer neural networks can approximate a variety of function classes arbitrarily well [13, 30]. For this reason, they are often coined the term *universal approximators* [7] and make a compelling choice for solving the problem of mapping time-frequency representations of noisy speech to the corresponding ideal binary masks. In the following section, we introduce a special class of feed-forward neural networks.

3.1.2 Convolutional Neural Networks

Convolutional neural networks (CNNs) are a specialized type of neural networks which utilize the convolution operator in at least one of their layers [19, Ch. 9]. They were originally proposed to overcome some of the limitations of fully-connected DNNs when dealing with image data, such as the topology of the input data not being utilized and no built-in invariance in respect to translations or local distortions of the input data [39]. In addition to image data, CNNs are also commonly used for other types of data having a known grid-like topology, for example time series [11] or spectral representations of speech [75]. In this section we specifically focus on two-dimensional spectrogram data, although similar formulations exist for data of other dimensions.

In a CNN, the neurons are arranged into *feature maps* in which the neurons share the same parameters. Instead of being fully-connected, each neuron is connected only to a small neighbourhood of the previous layer, named the *receptive field* of the neuron. The neurons in a feature map all perform the same operation on the layer input: a convolution between a learned *kernel* and their respective receptive field. This is done by essentially shifting a small context window sequentially over the input data, thereby computing a set of outputs. The local connectivity of the neurons is one of the key features in obtaining shift-invariance of recognizable patterns. For example, may be desirable to recognize a certain energy pattern in a spectrogram regardless of its position in time. By using local receptive fields, the neurons learn to detect low-level features, and by shifting the kernel across the input data, features are detected regardless of position [39, Sec. 2][71].

Following the notation of (3.3), the neurons in a feature map each compute

$$x_{p,q}^{(j)} = h^{(j)} \left(\sum_u \sum_v w_{p+u,q+v}^{(j)} x_{p,q}^{(j-1)} \right), \tag{3.5}$$

where $x_{p,q}^{(j)}$ is the p, q'th output of the j'th layer [19, Ch. 9]. Note that there are different ways of handling the convolutions near the edges of the input data. For example, one can choose to exclude the neurons (3.5) which use invalid indices in the summations. This leads to a reduction in dimensions for the resulting feature map. Another way is to zero-pad the input data accordingly, such that the dimensions are preserved.

Convolutional layers sometimes also contain *pooling* operations, which can further reduce the dimensions of the internal representations. Pooling serves to reduce the dimensions of the feature maps by combining multiple outputs from a region into a single output. Other than downsampling the input data, the pooling layer further promotes invariance to small shifts and distortions. This invariance is particularly useful in image recognition, where the position of a detected feature is important only in relation to other features and not in the image as a whole [39, Sec. 2]. For more on pooling, see [19, Ch. 9].

We end this section with some common practices regarding convolutional layers. Since each kernel can be interpreted as a *feature extractor*, it is common to use multiple kernels in each convolutional layer, resulting in collections of two-dimensional feature maps [7, Sec. 5.5]. Furthermore, due to small receptive fields, an overview of the entire input data is lost. This may complicate learning of higher-level features over larger regions. A common way to prevent this is by sequentially stacking multiple convolutional layers, such that the effective receptive fields of the neurons are progressively increased. Finally, convolutional layers are commonly placed earlier in the chain of transformations to act as feature extractors for e.g. fully-connected layers deeper into the network. In the following section, we motivate and discuss the use of CNNs for solving the problem at hand.

3.2 Model Design Choices

In order to successfully apply the beamformers for own-voice retrieval introduced in the previous chapter, we need a classifier capable of discriminating between target speech-dominated and noise-dominated time-frequency units. We choose to employ a CNN, which is a suitable model for numerous reasons:

- CNNs are representative learning methods [41], which are capable of learning discriminative features of complicated signals such as speech.
- Parameter-sharing means a smaller memory footprint, which is desirable for worn devices like hearing aids.

- Time-frequency representations of speech tend to have a strong local structure, where coefficients are highly correlated with the adjacent coefficients.
- CNNs are robust to translations of the input, which aids generalization across different speakers and speaking-styles.

Other reasonable choices of machine learning models include recurrent neural networks (RNN). These models are designed to process sequential data and capture temporal context [19, Ch. 10], and since audio is inherently serial these could be very valuable in modelling longer temporal dependencies. One variant of RNNs in particular, the *bi-directional long short-term memory* (BLSTM) network, has successfully been applied to solve various speech enhancement tasks [17, 27, 64]. In future work, the use of RNNs or perhaps a combined CNN/RNN approach utilizing the advantage of both models could be investigated.

The remaining of this chapter covers the design and hyperparameter optimization of the proposed CNN. As more layers are added to the network, the hyperparameter search space quickly becomes too vast for exhaustive searches to be practical. Therefore we choose to only investigate certain hyperparameters, while the remaining are chosen based on presumptions or preliminary experiments not reported here. In the following sections we discuss and motivate some of these presumptions, while experiments are conducted in Sec. 3.3.

Network Input and Training Targets

Although there exists many high-level acoustic features showing superior performance in various audio processing tasks [75], we choose to simply use the log-magnitude STFT coefficients as input to the CNN. The reason for this is three-fold: First of all, the beamformer operates in the time-frequency domain. Thus using non-invertible transformations like e.g. mel-frequency transformations would require a problematic inverse transformation - either learned internally in the trained network or as a post-processing step - in order for the estimated mask to be usable. Secondly, we expect the CNN to be capable of learning the relevant representations itself which are needed for discrimination. Finally, it should also be noted that magnitude spectrograms do not carry phase information. And although phase potentially is a discriminative feature, the structure in phase spectrograms is less clear than that of log-magnitude spectrograms (see Fig. 3.2). Therefore, we presume it is difficult for the network to learn discriminative features from operating directly on the phase spectrograms. The purpose of the logarithm is to compress the amplitudes, which serves to partially equalize the importance of loud and quieter sounds. For these reasons we choose the commonly used and perceptually motivated log-magnitude spectrograms.

Since speech signals are inherently serial, knowledge of past observations can aid the prediction of the future due to short-term correlations. In order to incorporate

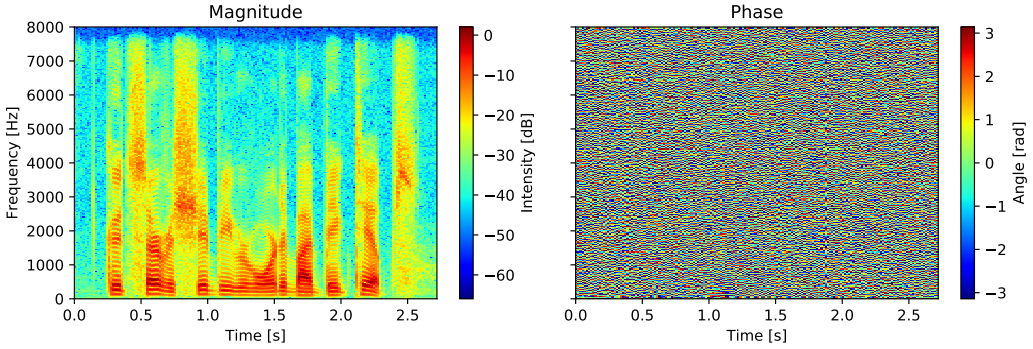


Figure 3.2: Magnitude and phase spectrograms of an utterance. The structure in the phase spectrogram may seem less obvious.

temporal context in the model, the input will consist of multiple consecutive time-frequency frames. The number of time frames is specified by the parameter l_{context}, which is named the *context window*. To reduce the algorithmic delay and enable online processing, the ideal mask is estimated for one time frame at a time. This means that the binary mask of frame index l is estimated using the matrix

$$\underline{\boldsymbol{Y}}(l) = [\boldsymbol{Y}(l - l_{\text{context}} - 1), \boldsymbol{Y}(l - l_{\text{context}} - 2), \ldots, \boldsymbol{Y}(l)] \in \mathbb{C}^{K \times l_{\text{context}}}, \tag{3.6}$$

where

$$\boldsymbol{Y}(l) = [Y(K, l), Y(K-1, l), \ldots, Y(0, l)]^\top \in \mathbb{C}^K,$$

omitting the logarithm and magnitude for a moment. Future frames can also be used, however doing so will introduce a delay in the system for it to remain causal. Specifically for hearing assistive devices, delays exceeding $20-30$ ms were found to be disruptive or disturbing due to asynchronicity in the sensory stimuli [66] (e.g. a perceptually noticeable delay between the aid-conducted sound and the bone-conducted sound or the visual stimuli). In an attempt to keep the delay low, the temporal context will solely consist of past frames. In literature [28, 43, 50], a temporal context of $100-700$ ms has commonly been used, and inspired by this we set $l_{\text{context}} = 7$ context frames, which corresponds to 128 ms of audio. In Figure 3.3, a context window is depicted.

In order to make the CNN fully utilize the microphone array, we need a way of incorporating the information from the secondary microphone in the mask estimation process. One way of doing this is by processing the channels jointly by expanding the dimensions of the input - similar to how color channels are stacked in image data. Alternatively, the channels can be processed separately using a generic network, resulting in M masks for M channels. The masks can then be combined into a single mask using e.g. the mean or median operator, as was proposed in [27]. Since we only consider $M = 2$ channels, we choose to explore methods of jointly processing

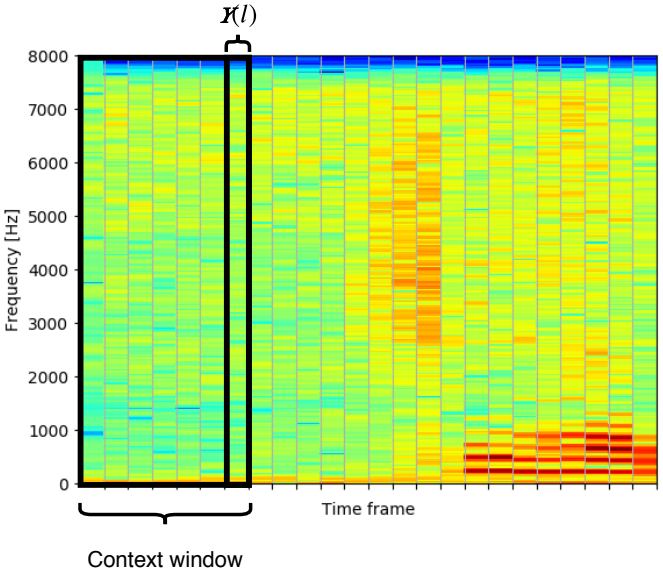


Figure 3.3: The input of the CNN consists of a concatenation of the current frame with l_{context} past frames, resulting in a $K \times l_{\text{context}}$ matrix. In the figure, $l = 7$ and $l_{\text{context}} = 7$.

the channels.

Three sets of input features of increasing sizes are compared. The first set consists of the reference channel only, and serves as a computationally lighter set. On a hearing aid, the reference microphone is likely to be the frontmost microphone, which has the shortest distance to the mouth and thus has the highest SNR. Thus spatial information is completely disregarded, which effectively makes the network monaural. In the second feature set, the secondary channel is included by stacking the channels along a third dimension. The inclusion of the secondary channel is expected to improve performance, but comes at the cost of increasing the number of required computations in the input layer. The final feature set was proposed in [68] as a way to incorporate phase correlations between the channels without explicitly passing the phase information. With a slight abuse of notation, let $|\underline{\boldsymbol{Y}}_m(l)|$ denote a magnitude spectrum excerpt of the m'th channel, as defined in (3.3). The authors suggested using the magnitude spectra of the channels (i.e. $|\underline{\boldsymbol{Y}}_1(l)|$ and $|\underline{\boldsymbol{Y}}_2(l)|$) combined with the magnitude *sum* and *difference* of the channel spectra (i.e. $|\underline{\boldsymbol{Y}}_1(l)+\underline{\boldsymbol{Y}}_2(l)|$ and $|\underline{\boldsymbol{Y}}_1(l)-\underline{\boldsymbol{Y}}_2(l)|$). These spectra are then stacked to what essentially becomes a four-channel input. In Sec. 3.3, experiments are carried out to determine the potential performance increase in relation to the computational requirements of the three feature sets.

3.2.1 Choosing an Architecture

The proposed network is depicted in Fig. 3.4 and will consist of a number of convolutional layers (see Eq. (3.5)), followed by a number of fully-connected layers (see Eq. (3.3)). Here, the convolutional blocks are meant as feature extractors, whereas the fully-connected block acts as the classifier. Unless otherwise stated, the convolutional operations are sufficiently zero-padded at the edges such that the dimensions of the input is preserved.

Although pooling layers are a fundamental part of CNNs [41], they were not improving performance of the network configurations considered in this study. For this reason pooling is not used.

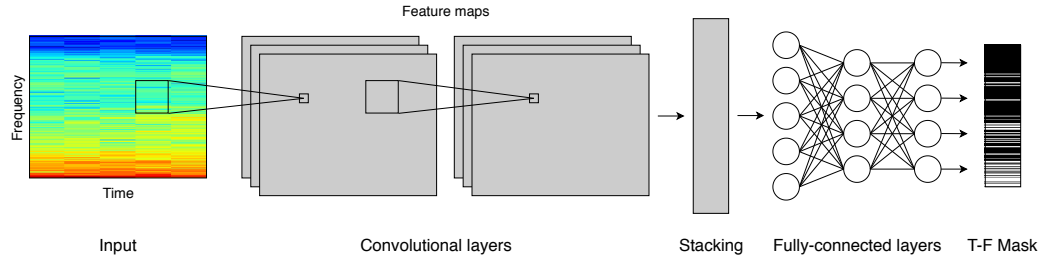


Figure 3.4: An overview of the proposed architecture.

Activation Functions and Optimization Algorithms

Variants of the rectified linear unit (ReLU) activation functions have risen in popularity in the recent years to the point where the use of sigmoidal functions is almost obsolete [19]. The authors in [32] found that the use of rectified activation functions is the single most important factor for improving accuracy in classification tasks. For these reasons, we choose to use the ReLU activation functions in the hidden layers, but choose to use a sigmoid function in the output layer to bound the prediction to the interval $[0, 1]$.

Regarding optimization algorithms, preliminary experiments were carried out using various set of parameters for the algorithms: Adam [34], stochastic gradient descent [58] and Adadelta [80]. Among the three, Adam led to the highest network accuracies in about half the convergence time of the other two algorithms. For this reason, Adam is chosen as the optimization algorithm in the remaining sections.

Regularization

To improve the generalization capabilities of the network, we will utilize *dropout* during training [29]. Dropout introduces perturbations to the network by randomly excluding a certain percentage of the network for each training example. By doing so, complex co-adaptations are prevented from being learned, where some neurons are only helpful in the context of certain other neurons. Instead, each neuron learns to detect a feature which is more generally helpful for producing the correct answer. Dropout has been shown to significantly reduce overfitting on smaller datasets and improve performance in a variety of supervised learning tasks [29].

Another recently proposed method is *batch normalization* [31]. Batch normalization can be employed to mitigate changes in the input distributions to the individual layers as learning takes place. Specifically, as parameters change in a layer, so will the input distributions of the preceding layers. This change in distribution presents a problem because the layers need to adapt to continuously changing distributions [31]. In order to mitigate this, it is proposed to normalize the propagated data vector before each non-linearity using running estimates of the mean and standard deviation of the input distribution in each layer. In addition to normalization, an affine transformation with learned parameters is applied subsequently to ensure that the inserted transformations can represent the identity transform (i.e. the normalization operation can be inverted if needed).

Batch normalization has been shown to largely improve training speed by allowing higher learning rates, and in some cases also improved performance [31]. For these reasons, we choose to utilize batch normalization.

3.2.2 Selecting a Kernel Size

In image processing literature [81], small and square kernels are commonly used (e.g. 3×3 or 5×5) with the intention that the first convolutional layers should learn low-level features like edges and corners. The kernels are symmetric, because the axes represent similar things (i.e. position).

However, convolutional layers designed to extract relevant features for image processing may not transfer well to other types of data, such as audio spectrograms. In fact, the design choices are suboptimal for a couple of reasons:

Different Meaning of Axes

For regular images, the kernel dimensions carry the same meaning. The same is not true for spectrograms, where the axes represent *time* and *frequency*. Thus there is little reason to believe that symmetric kernels of small dimensions are well-suited for

time-frequency representations of audio. For example, wider kernels are capable of learning longer temporal dependencies, since more temporal information is available. And on the other hand, using taller kernels may better capture spectral patterns of speech [53, 54]. In the extreme cases, one may choose to do one-dimensional convolutions only, e.g. only along the time axis. Note that although a convolutional layer is incapable of learning frequency dependencies this way, layers deeper in the network are still capable of this by combining outputs from the neurons. Accordingly, non-symmetric kernel shapes are included in the search space when optimizing kernel sizes.

Sounds Are Transparent

Most pixels in image data can be attributed to a single object. In spectrograms, however, each coefficient represents the sum of energy at that specific time and frequency. Hence multiple audio sources may contribute to a single energy density and we only observe the accumulated effect, including any phase cancellations that might occur. Due to this, it may be difficult to accurately detect and discriminate between audio sources occupying the same time-frequency tiles. Furthermore, many sources of audio encountered in the real world are *wideband*, meaning that the signal energy is distributed over a large range of frequencies. For example, the harmonics of voiced speech are regularly spaced in frequency. And as a sound is raised in pitch, not only are the harmonics shifted in frequency, but so is the spatial extend [59, 79]. In order to better handle these issues, we choose to explore network architectures having several convolutional layers with larger kernels and many feature maps.

3.3 Hyperparameter Experiments

In this section, experiments are carried out for optimizing the following hyperparameters of the CNN: *number of convolutional layers*, *size of convolutional kernels* and *choice of input features*. Performance will be assessed using the validation set error as metric.

To speed up the parameter search, a reduced training set is considered, corresponding to 90000 training examples or approximately 190 minutes. It is then presumed that a well-performing network architecture found this way is going to perform similarly when trained on the full dataset. This is assuming the architecture has the learning *capacity* to accommodate the increased dataset size, such that the model does not underfit.

The noisy speech signals are synthesized according to App. A.1. The training and validation set are generated collectively and then split into the respective sets using a 87.5%-12.5% segmentation. The input features are generated by concatenating con-

Num. of layers	Validation loss	Num. of parameters (M)
2	0.195	1.061
3	0.188	1.262
4	0.186	1.463
5	0.180	1.664
6	0.181	1.865
7	0.180	2.066

Table 3.1: Validation set error for various number of convolutional layers.

	Height				
Width	3	5	7	9	11
3	0.184	0.185	0.186	0.180	0.185
5	0.187	0.179	0.181	0.185	0.184
7	0.185	0.181	0.181	0.179	0.182

Table 3.2: Validation set error for various kernel sizes. The axes denote width (along the time axis) and height (along the frequency axis).

secutive TF units according to (3.6) followed by taking the element-wise magnitude and logarithm. Before applying the logarithm, a small constant is added to avoid excessively small magnitudes.

The network is trained by minimizing the binary cross-entropy (see Eq. (3.4)) between the network output and the ideal binary mask. Network training will be terminated when the validation set error or starts to decline, known as *early stopping*. This is implemented to prevent overfitting. Specifically, if the validation set error has not improved in the ten most recent epochs, training stops and the model is rolled back to the best performing epoch. Finally, the hyperparameter configuration resulting in the lowest validation set error is selected as the final model.

In the first experiment, six networks are trained with increasing number of layers. The results are shown in Table 3.1. To keep the memory requirements low, we choose to use five convolutional layers. In this experiment, the kernel sizes of the convolutional layers are determined. The experiment is motivated by the discussion in Sec. 3.2.2. Note that since the input data consists of 7 time-frames, we do not investigate kernel sizes above 7 along the time axis. The results are shown in Table 3.2. Here, the difference between the best performing kernel sizes are negligible (≤ 0.002), hence we choose the smallest among these which is 5×5. In the final experiment we investigate different types of input feature sets. The results are shown in Table 3.3. Here, 'Ref channel only' represents the monaural feature set, 'Two-channel' represents the two-channel feature set and 'Two-channel + inter-channel' represents the

Type of input	Validation loss	FLOPS (G)
Ref. channel only	0.184	585.628
Two-channel	0.165	588.622
Two-channel + inter-channel	0.150	594.609

Table 3.3: Validation loss and floating-point operations per second (FLOPS) for three sets of input features. The sets correspond to the ones discussed in Sec. 3.2.

four-channel set.

By examining Table 3.3, the validation loss improves considerably when introducing the secondary channel, and a further reduction is seen when including the inter-channel features. Furthermore, the increase in FLOPS by processing additional channels is negligible ($\approx 0.5\%$ and $\approx 1.5\%$ for the models, respectively). For these reasons, we choose to use the four-channel feature set combining intra-channel and inter-channel features. The proposed architecture is summarized in Table .3.4.

Layer	Configuration	Output dimensions
Convolution Batch Normalization ReLU	7×7 @ 64	$257 \times 7 \times 64$
Convolution Batch Normalization ReLU	7×7 @ 64	$257 \times 7 \times 64$
Convolution Batch Normalization ReLU	7×7 @ 64	$257 \times 7 \times 64$
Convolution Batch Normalization ReLU	7×7 @ 64	$257 \times 7 \times 64$
Convolution Batch Normalization ReLU	7×7 @ 64	$257 \times 7 \times 64$
Frame stacking		1799
Fully-connected ReLU	1799 units	512
Fully-connected Sigmoid	512 units	257

Table 3.4: Architecture of the proposed CNN. Dropout is applied to all hidden layers with a probability of 0.3.

Chapter 4

System Evaluation and Results

In this chapter, numerical experiments are conducted to evaluate the system performance. First, an analysis of the classification capabilities of the proposed CNN is given. Then the full own-voice retrieval system is assessed and results are presented.

4.1 Receiver Operating Characteristic Analysis

Although the target mask is binary, the output of the DNN will be a *soft* mask consisting of numeric values in the range of $[0, 1]$. The entries of this mask can be interpreted as specifying to which degree each TF unit is speech-dominated - as deemed by the classifier. In order to map these values to decisions (i.e. whether to update the noise covariance matrix or not), we need to specify what comprises a *speech-dominated* and *noise-dominated* unit. Assuming the classifier is more likely to assign high values to speech-dominated units than noise-dominated units, a *cut-off value* $c \in [0, 1]$ can be defined to determine a decision rule as follows: units with values below c are predicted as being noise-dominated, and units with values above c are predicted as being speech-dominated. Selecting a cut-off value is a trade-off: setting c too small increases the rate of correctly classifying speech-dominated time-frequency units, while decreasing the rate of correctly classifying noise-dominated units. Setting c too large accomplishes the opposite. These quantities are often called the *true positive rate* (TPR) and *true negative rate* (TNR). Ideally, the cut-off value is optimal in some sense, e.g. it maximizes the rate of successful classification.

A *receiver operating characteristic* (ROC) curve is a two-dimensional depiction of classifier performance, in which the *true positive rate* (TPR) is plotted as a function of the *false positive rate* (FPR). The FPR represents the ratio of correctly classified negative (i.e. noise-dominated) TF units to the total number of TF units [69]. The ROC curve depicts the relative trade-off between true positives and false positives. AUC refers to the *area under the curve*, a statistic commonly used for summarizing ROC curves[69]. The ROC curve for the proposed DNN is shown in Fig. 4.1.

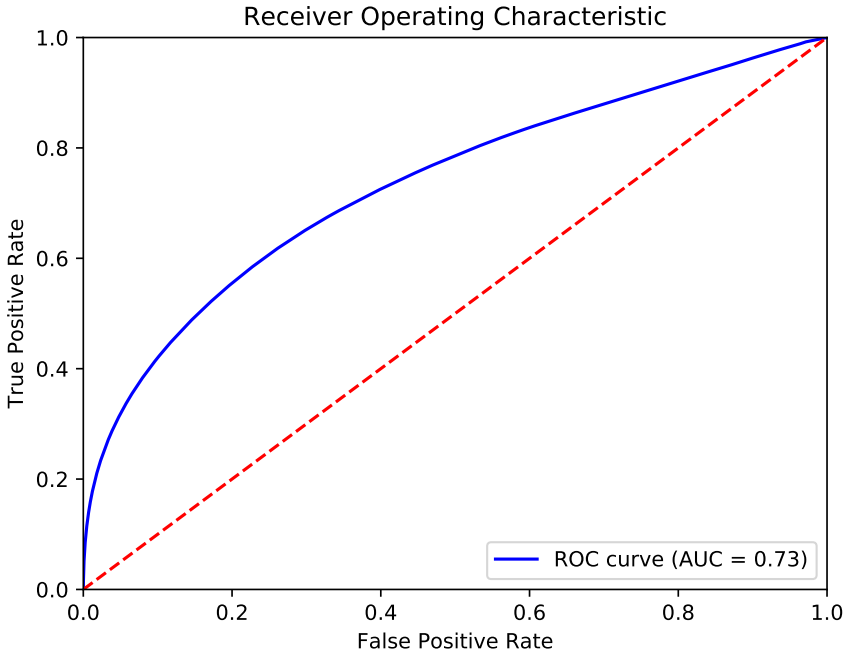


Figure 4.1: Receiver operating characteristic (ROC) curve of the DNN. The dashed red line is the *chance* line where tpr = fpr.

If the *cost* of wrongly classifying speech- and noise-dominated units is known, as well as the prevalence of either case, a cost function can be formulated, from which a statistical optimal cutoff value can be determined [69]. The cost and prevalence quantities are, however, unknown and are beyond the scope of this thesis. Instead, what is commonly done [51] is to maximize *Youden's J statistic* defined as

$$J = \max_c \text{TPR} + \text{TNR} - 1. \tag{4.1}$$

This equates to $J = 0.409$, hence $c = 0.409$ is chosen for the remaining of this chapter.

4.2 System Evaluation

In this section the performance of the full own-voice retrieval system is assessed using the performance measures described in Sec. 1.2. An overview of the system is shown in Figure 4.2.

It should be noted that the amount of information retained in the IBM is dependent on the SNR of the mixture. For example, a mixture with very low SNR will have few time-frequency tiles with desirable local SNRs, whereas a high SNR mixture will

primarily consist of speech-dominated time-frequency tiles. Because of this, the LC should be chosen in relation to the SNR. In [35], it is instead suggested to use the *relative criterion*, defined as the difference between the LC and SNR, which makes the IBM invariant to changes in SNR. Inspired by this, we choose a relative criterion of - 3 dB in order to preserve additional speech information.

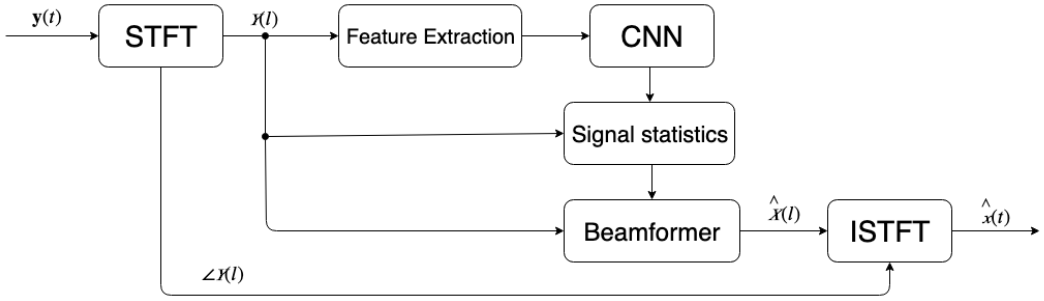


Figure 4.2: System overview. Noisy observations are STFT-transformed and features are extracted. From these, a CNN estimated the IBM, which in turn is used to update the covariance matrix estimates. A beamformer solution is then computed and applied, and the filtered signal is inverse STFT-transformed using the noisy phase.

Performance across noise types

As a first step, we examine performance across the noise types described in App. A.1. The results are shown in Fig. 4.3 and 4.4 for the MVDR and MWF beamformer, respectively. Comparing Fig. 4.3 with Fig. 4.4, it is seen that the oracle MVDR beamformer achieves a considerably higher PESQ score across all noise types. The oracle MWF beamformer, however, achieves a much higher Seg-SNR. When examining the performance of the estimated mask, it is seen that the MVDR beamformer improves the scores for PESQ, STOI and ESTOI across all noise types, except for `ssn` which remains mostly unaltered. The same is not true for the MWF beamformer, where STOI and ESTOI scores are worse than the unprocessed signal, and scores are mostly unaltered for `ssn` and `str`. Additionally, the PESQ score remains the same for `bus` and `ped`. Ignoring Seg-SNR for a moment, the largest improvements are seen in ESTOI and PESQ for `bbl` and `caf`.

Performance across SNRs

Next, we examine performance across SNRs. The results are shown in Fig. 4.5 and 4.6. Looking at PESQ first, neither beamformer are capable of matching the oracle performance for high SNR. Additionally, slightly higher PESQ score is achieved by the MVDR beamformer. Regarding STOI and ESTOI, the MVDR bemformer achieves scores near that of the oracle MVDR. The MWF, on the other hand, only

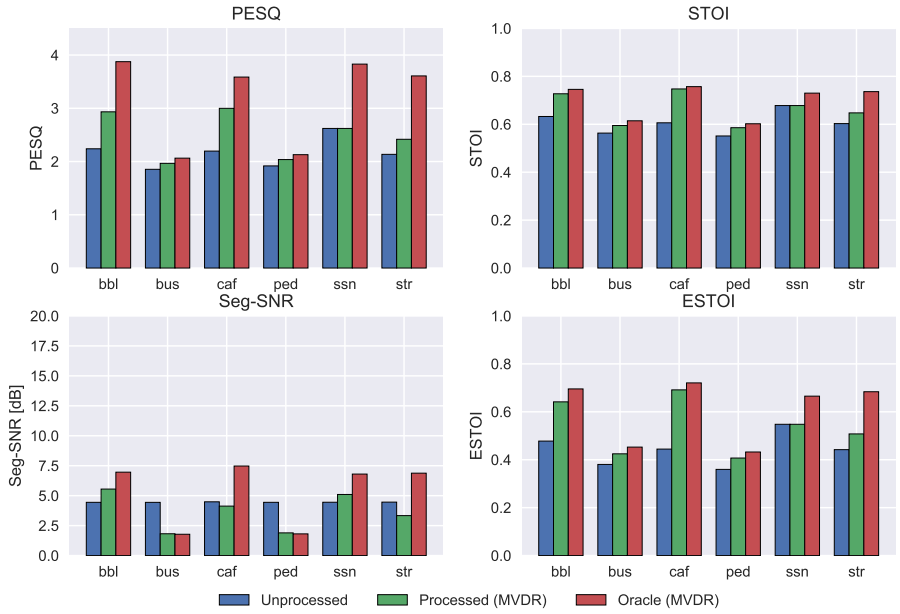


Figure 4.3: Results for full system using MVDR for various noise types.

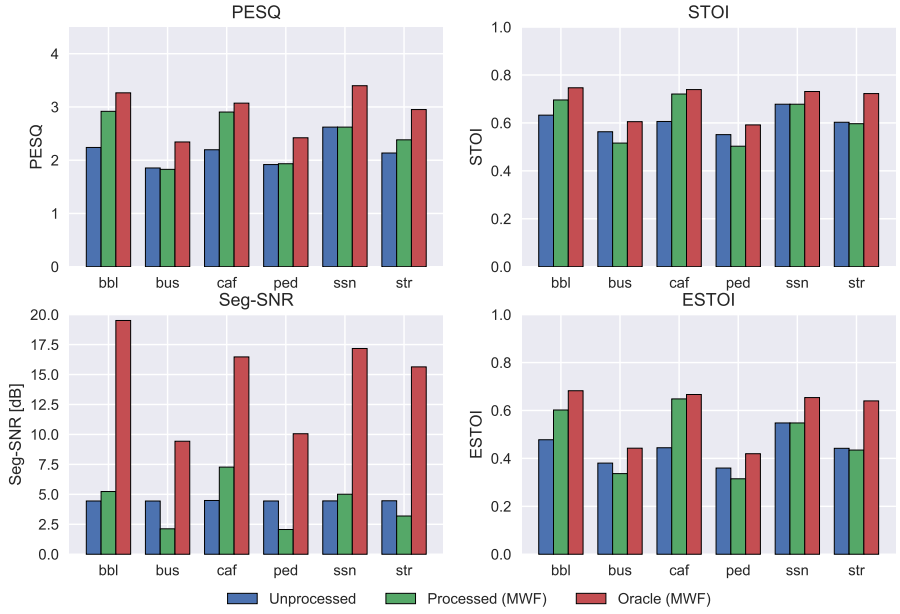


Figure 4.4: Results for full system using MWF for various noise types.

performs well in low SNRs, and degrades the signals below the score of the un-

processed signals for high SNRs. Finally, the Seg-SNR only improves in low SNRs and falls below the unprocessed Seg-SNR for high SNR. Specifically for the MWF beamformer, the Seg-SNR seems constant across SNRs.

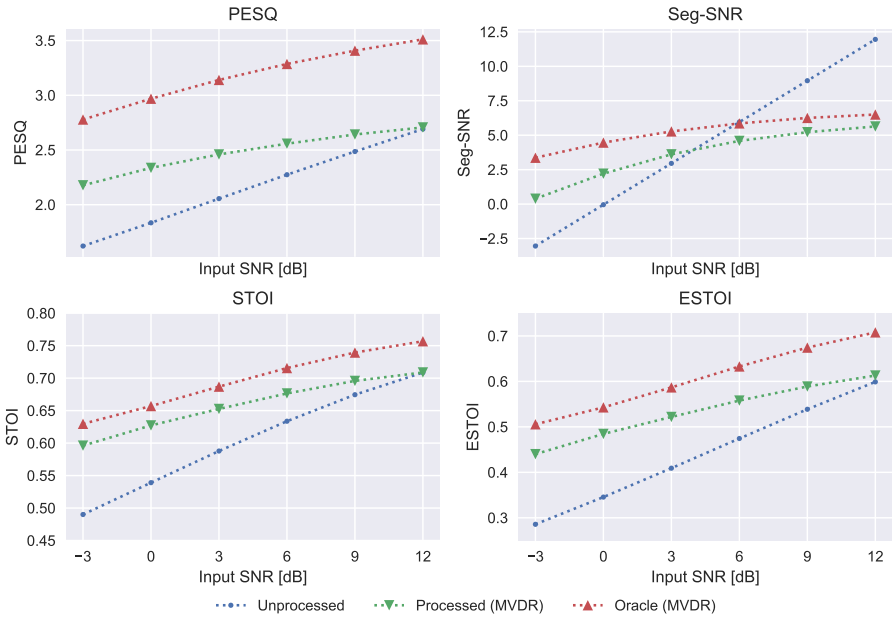


Figure 4.5: Results for full system using MVDR for various SNRs.

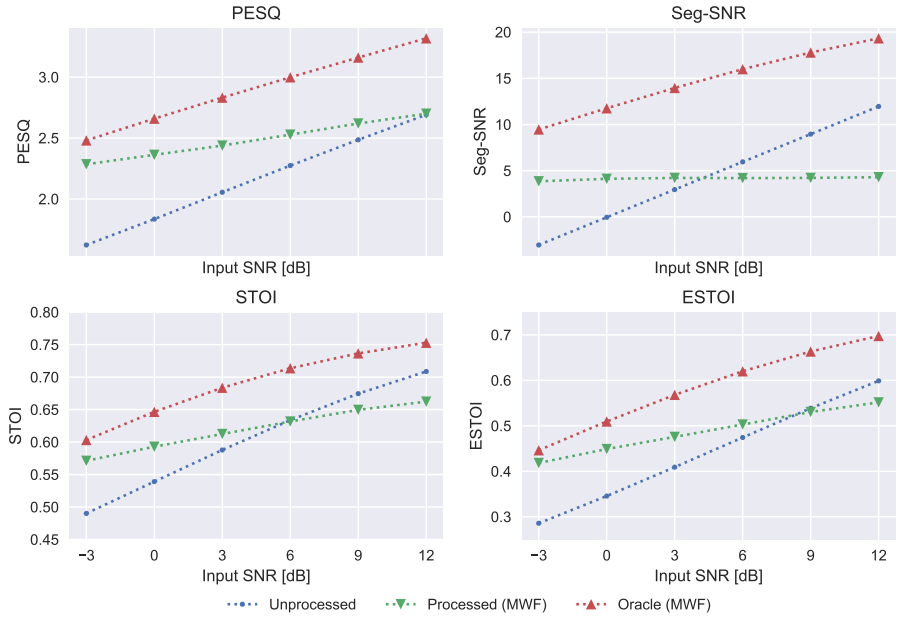


Figure 4.6: Results for full system using MWF for various SNRs.

Conclusion

In this thesis, a deep learning-based beamformer has been implemented and evaluated for own-voice retrieval in additive background noise. The core of the own-voice retrieval system is the beamforming system, which relies on accurate estimates of the spatial covariance matrices of the noise and own-voice signals. Inspired by the recent success of RNNs and CNNs in estimating ideal masks, a CNN is employed to estimate the ideal binary mask from noisy time-frequency features, which in turn is used in the computation and continually tracking of the signal statistics required for beamforming. We show that the MVDR and MWF beamformer depend solely on the spatial covariance matrices.

Although a monaural CNN is capable of estimating ideal binary masks, the inclusion of the secondary channel improved performance considerably. Additionally, when including inter-channel features, further improvement was seen. This indicates that the CNN is capable of utilizing multi-channel input for more accurate estimation.

The results show considerable improvements in terms of PESQ, STOI and ESTOI for selected noise types. The challenging noise types were babble noise and diffuse speech-shaped noise. Even for low SNRs (-3 to 0 dB) there were substantial improvements. For high SNRs (6 to 9dB), the gains were smaller. Comparing the MWF and MVDR beamformer, the MVDR came out ahead as being more robust to noise type and SNR.

Part of the goal was to find a memory- and computationally efficient method, which can be implemented on future embedded systems such as hearing aids. Arguably, the CNN is quite inefficient with 1.371 million parameters and 594.608 MFLOPS compared to other non-deep learning approaches. However, with the increasing computational capabilities of hearing assistive devices, the use of CNNs may be feasible solutions in the future.

As future work, one could explore methods for reducing the memory and computational requirements by e.g. exploring shallow architectures. Additionally, since the majority of the parameters are occupying the fully-connected layers, methods such as *parameter pruning* can potentially reduce the model size tremendously by removing

unimportant parameters. For example, [70] reported a 94.72% reduction in model size for a generic three-layer DNN trained to classify the MNIST [40] dataset before the accuracy drops below 1%.

Bibliography

[1] Raviraj Adve. Notes: Optimal beamforming, 2007. URL https://www.comm.utoronto.ca/~rsadve/Notes/BeamForming.pdf. [Cited on page 11]

[2] Jon Barker, Ricard Marxer, Emmanuel Vincent, and Shinji Watanabe. The third chime speech separation and recognition challenge: Analysis and outcomes. *Computer Speech and Language*, 46:605–626, 2017. [Cited on pages 1 and 50]

[3] Jon Barker, Shinji Watanabe, Emmanuel Vincent, and Jan Trmal. The fifth chime speech separation and recognition challenge: Dataset, task and baselines. In *Proceedings of the 19th Annual Conference of the International Speech Communication Association (INTERSPEECH 2018)*, Hyderabad, India, September 2018. [Cited on page 1]

[4] Jacob Benesty, Jingdong Chen, Yiteng Huang, and Israel Cohen. *Noise Reduction in Speech Processing*. Springer Publishing Company, Incorporated, 1st edition, 2009. ISBN 3642002951, 9783642002953. [Cited on pages 1, 6, and 16]

[5] Jacob Benesty, Shoji Makino, and Jingdong Chen. *Speech Enhancement*. Springer Publishing Company, Incorporated, 1st edition, 2010. ISBN 3642063179, 9783642063176. [Cited on page 9]

[6] Jacob Benesty, Jingdong Chen, and Emanul A.P. Habets. *Speech Enhancement in the STFT Domain*. Springer Publishing Company, Incorporated, 1st edition, 2012. ISBN 3642232493, 9783642232497. [Cited on pages 6 and 7]

[7] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Information Science and Statistics. Springer, 5th edition, 2006. ISBN 9780387310732. [Cited on pages 20, 21, 22, and 23]

[8] H. Bölcskei, P. Grohs, G. Kutyniok, and P. Petersen. Optimal approximation with sparsely connected deep neural networks. *SIAM Journal on Mathematics of Data Science*, 1(1):8–45, 2019. doi: 10.1137/18M118709X. [Cited on page 20]

[9] M. Brandstein and D. Ward. *Microphone Arrays: Signal Processing Techniques and Applications*. Digital Signal Processing - Springer-Verlag. Springer, 2001. ISBN 9783540419532. [Cited on pages 1, 5, and 13]

[10] A.S. Bregman. *Auditory Scene Analysis: The Perceptual Organization of Sound*. Bradford book. MIT Press, 1994. ISBN 9780262521956. [Cited on page 2]

[11] J. Chen, W. Chen, C. Huang, S. Huang, and A. Chen. Financial time-series data analysis using deep convolutional neural networks. In *2016 7th International Conference on Cloud Computing and Big Data (CCBD)*, pages 87–92, Nov 2016. doi: 10.1109/CCBD.2016.027. [Cited on page 22]

[12] L Cremer and H A. Muller. Principles and applications of room acoustics i. *Physics Today*, 37, 01 1984. doi: 10.1063/1.2916055. [Cited on page 5]

[13] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4):303–314, Dec 1989. ISSN 1435-568X. doi: 10.1007/BF02551274. [Cited on page 22]

[14] S. Doclo and M. Moonen. Gsvd-based optimal filtering for single and multimicrophone speech enhancement. *IEEE Transactions on Signal Processing*, 50(9): 2230–2244, Sept 2002. ISSN 1053-587X. doi: 10.1109/TSP.2002.801937. [Cited on page 12]

[15] Simon Doclo, Ann Spriet, Jan Wouters, and Marc Moonen. *Speech Distortion Weighted Multichannel Wiener Filtering Techniques for Noise Reduction*, pages 199–228. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005. ISBN 978-3-540-27489-6. doi: 10.1007/3-540-27489-8_9. [Cited on page 1]

[16] Simon Doclo, Sharon Gannot, Marc Moonen, and Ann Spriet, editors. *Acoustic Beamforming for Hearing Aid Applications*, pages 269–302. 2010. ISBN 978-0-470-37176-3. doi: 10.1002/9780470487068.ch9. [Cited on page 11]

[17] H. Erdogan, J. R. Hershey, S. Watanabe, and J. Le Roux. Phase-sensitive and recognition-boosted speech separation using deep recurrent neural networks. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 708–712, April 2015. doi: 10.1109/ICASSP.2015.7178061. [Cited on page 24]

[18] G.B. Folland. *Fourier Analysis and Its Applications*. Pure and applied undergraduate texts. American Mathematical Society, 2009. ISBN 9780821847909. [Cited on page 6]

[19] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org. [Cited on pages 19, 20, 21, 22, 23, 24, and 27]

[20] Karlheinz Gröchenig. *Foundations of Time–Frequency Analysis*. 01 2001. doi: 10.1007/978-1-4612-0003-1. [Cited on page 7]

[21] E. A. P. Habets, J. Benesty, I. Cohen, S. Gannot, and J. Dmochowski. New insights into the MVDR beamformer in room acoustics. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(1):158–170, Jan 2010. ISSN 1558-7916. doi: 10.1109/TASL.2009.2024731. [Cited on page 11]

[22] Emanuël Habets, Jacob Benesty, Sharon Gannot, and Israel Cohen. *The MVDR Beamformer for Speech Enhancement*, pages 225–254. 12 2009. doi: 10.1007/978-3-642-11130-3_9. [Cited on page 11]

[23] John H. L. Hansen and Bryan L. Pellom. An effective quality evaluation protocol for speech enhancement algorithms. In *ICSLP*, 1998. [Cited on page 8]

[24] Jens Heitkaemper, Jahn Heymann, and Reinhold Haeb-Umbach. Smoothing along frequency in online neural network supported acoustic beamforming. 10 2018. [Cited on page 16]

[25] Richard C. Hendriks, Timo Gerkmann, and Jesper Jensen. *DFT-Domain Based Single-Microphone Noise Reduction for Speech Enhancement: A Survey of the State of the Art*. Morgan & Claypool, 2013. URL https://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6813348. [Cited on pages 6 and 16]

[26] L. Hertel, H. Phan, and A. Mertins. Comparing time and frequency domain for audio event recognition using deep learning. In *2016 International Joint Conference on Neural Networks (IJCNN)*, pages 3407–3411, July 2016. doi: 10.1109/IJCNN.2016.7727635. [Cited on page 22]

[27] J. Heymann, L. Drude, A. Chinaev, and R. Haeb-Umbach. Blstm supported gev beamformer front-end for the 3rd chime challenge. In *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pages 444–451, Dec 2015. doi: 10.1109/ASRU.2015.7404829. [Cited on pages 24 and 25]

[28] Jahn Heymann, Lukas Drude, and Reinhold Haeb-Umbach. A generic neural acoustic beamforming architecture for robust multi-channel speech processing. *Computer Speech and Language*, 46:374 – 385, 2017. ISSN 0885-2308. doi: https://doi.org/10.1016/j.csl.2016.11.007. [Cited on page 25]

[29] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580, 2012. URL http://arxiv.org/abs/1207.0580. [Cited on page 28]

[30] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Netw.*, 2(5):359–366, July 1989. ISSN 0893-6080. doi: 10.1016/0893-6080(89)90020-8. [Cited on page 22]

[31] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015. [Cited on page 28]

[32] K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun. What is the best multi-stage architecture for object recognition? In *2009 IEEE 12th International Conference on Computer Vision*, pages 2146–2153, Sep. 2009. doi: 10.1109/ICCV.2009.5459469. [Cited on page 27]

[33] J. Jensen and C. H. Taal. An algorithm for predicting the intelligibility of speech masked by modulated noise maskers. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24(11):2009–2022, Nov 2016. ISSN 2329-9290. doi: 10.1109/TASLP.2016.2585878. [Cited on page 8]

[34] Diederik Kingma and Jimmy Ba. Adam: a method for stochastic optimization (2014). *arXiv preprint arXiv:1412.6980*, 15, 2015. [Cited on page 27]

[35] Ulrik Kjems, Jesper B. Boldt, Michael S. Pedersen, Thomas Lunner, and DeLiang Wang. Role of mask pattern in intelligibility of ideal binary-masked noisy speech. *The Journal of the Acoustical Society of America*, 126(3):1415–1426, 2009. doi: 10.1121/1.3179673. [Cited on page 35]

[36] M. Kolbœk, Z. Tan, and J. Jensen. Speech enhancement using long short-term memory based recurrent neural networks for noise robust speaker verification. In *2016 IEEE Spoken Language Technology Workshop (SLT)*, pages 305–311, Dec 2016. doi: 10.1109/SLT.2016.7846281. [Cited on page 50]

[37] H. Krim and M. Viberg. Two decades of array signal processing research: the parametric approach. *IEEE Signal Processing Magazine*, 13(4):67–94, July 1996. ISSN 1053-5888. doi: 10.1109/79.526899. [Cited on pages 9 and 10]

[38] H. Kuttruff. *Room Acoustics*. Taylor & Francis, fifth edition, 2009. [Cited on page 4]

[39] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, Nov 1998. ISSN 0018-9219. doi: 10.1109/5.726791. [Cited on pages 22 and 23]

[40] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. http://yann.lecun.com/exdb/mnist/. URL `http://yann.lecun.com/exdb/mnist/`. [Cited on page 40]

[41] Yann LeCun, Yoshua Bengio, and Geoffrey E. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015. doi: 10.1038/nature14539. URL `https://doi.org/10.1038/nature14539`. [Cited on pages 22, 23, and 27]

[42] Ning Li and Philipos C Loizou. Factors influencing intelligibility of ideal binary-masked speech: Implications for noise reduction. *The Journal of the Acoustical Society of America*, 123:1673–82, 04 2008. doi: 10.1121/1.2832617. [Cited on page 15]

[43] Y. Liu, A. Ganguly, K. Kamath, and T. Kristjansson. Neural network based time-frequency masking and steering vector estimation for two-channel mvdr beamforming. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6717–6721, April 2018. doi: 10.1109/ICASSP.2018.8462069. [Cited on pages 16 and 25]

[44] P.C. Loizou. *Speech Enhancement: Theory and Practice, Second Edition*. Taylor & Francis, 2013. ISBN 9781466504219. [Cited on page 8]

[45] Philipos C. Loizou. *Speech Quality Assessment*, pages 623–654. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011. ISBN 978-3-642-19551-8. doi: 10.1007/978-3-642-19551-8_23. [Cited on page 8]

[46] A. Narayanan and D. Wang. Ideal ratio mask estimation using deep neural networks for robust speech recognition. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 7092–7096, May 2013. doi: 10.1109/ICASSP.2013.6639038. [Cited on page 2]

[47] Emil Solsbæk Ottosen. *Sparse Nonstationary Gabor Expansions - with Applications to Music Signals*. PhD thesis, 2018. PhD supervisor: Prof. Morten Nielsen, Dept. of Mathematical Sciences, Aalborg University. [Cited on page 6]

[48] K. K. Paliwal, J. G. Lyons, and K. K. Wójcicki. Preference for 20-40 ms window duration in speech analysis. In *2010 4th International Conference on Signal Processing and Communication Systems*, pages 1–4, Dec 2010. [Cited on page 6]

[49] C. Pan, J. Chen, and J. Benesty. Performance study of the mvdr beamformer as a function of the source incidence angle. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 22(1):67–79, Jan 2014. ISSN 2329-9290. doi: 10.1109/TASL.2013.2283104. [Cited on page 50]

[50] Se Rim Park and Jinwon Lee. A fully convolutional neural network for speech enhancement. In *INTERSPEECH*, 2017. [Cited on page 25]

[51] Sara Perez-Jaume, Konstantina Skaltsa, Natàlia Pallarès, and Josep Carrasco. Thresholdroc : Optimum threshold estimation tools for continuous diagnostic tests in r. *Journal of Statistical Software*, 82, 11 2017. doi: 10.18637/jss.v082.i04. [Cited on page 34]

[52] K. B. Petersen and M. S. Pedersen. The matrix cookbook, nov 2012. URL `http://www2.imm.dtu.dk/pubdb/p.php?3274`. Version 20121115. [Cited on page 14]

[53] J. Pons and X. Serra. Designing efficient architectures for modeling temporal features with convolutional neural networks. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2472–2476, March 2017. doi: 10.1109/ICASSP.2017.7952601. [Cited on page 29]

[54] J. Pons, T. Lidy, and X. Serra. Experimenting with musically motivated convolutional neural networks. In *2016 14th International Workshop on Content-Based Multimedia Indexing (CBMI)*, pages 1–6, June 2016. doi: 10.1109/CBMI.2016.7500246. [Cited on page 29]

[55] V. Pulkki, S. Delikaris-Manias, and A. Politis. *Parametric Time-Frequency Domain Spatial Audio*. Wiley - IEEE. Wiley, 2017. ISBN 9781119252597. [Cited on page 16]

[56] Lawrence Rabiner and Biing-Hwang Juang. *Fundamentals of Speech Recognition*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1993. ISBN 0-13-015157-2. [Cited on pages 1 and 9]

[57] A. W. Rix, J. G. Beerends, M. P. Hollier, and A. P. Hekstra. Perceptual evaluation of speech quality (pesq)-a new method for speech quality assessment of telephone networks and codecs. In *Proceedings of the Acoustics, Speech, and Signal Processing, 200. On IEEE International Conference - Volume 02*, ICASSP 01, pages 749–752. IEEE Computer Society, 2001. ISBN 0-7803-7041-4. doi: 10.1109/ICASSP.2001.941023. [Cited on page 8]

[58] Herbert Robbins and Sutton Monro. A stochastic approximation method. *Ann. Math. Statist.*, 22(3):400–407, 09 1951. doi: 10.1214/aoms/1177729586. [Cited on page 27]

[59] Daniel Rothmann. What's wrong with cnns and spectrograms for audio processing? https://towardsdatascience.com/whats-wrong-with-spectrograms-and-cnns-for-audio-processing-311377d7ccd, 2018. Accessed: 2019-05-6. [Cited on page 29]

[60] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–, October 1986. [Cited on page 20]

[61] J S Garofolo, Lori Lamel, W M Fisher, Jonathan Fiscus, D S. Pallett, N L. Dahlgren, and V Zue. Timit acoustic-phonetic continuous speech corpus. *Linguistic Data Consortium*, 11 1992. [Cited on page 50]

[62] T. N. Sainath, A. Narayanan, R. J. Weiss, E. Variani, K. W. Wilson, M. Bacchiani, and I. Shafran. Reducing the computational complexity ofmultimicrophone acoustic models with integrated feature extraction. *inProc. Interspeech*, page 971–1975, 2016. [Cited on page 6]

[63] R.H. Shumway and D.S. Stoffer. *Time Series Analysis and Its Applications: With R Examples*. Springer Texts in Statistics. Springer New York, 2010. ISBN 9781441978646. [Cited on page 15]

[64] Peter Sibbern Frederiksen, Jesus Villalba, Shinji Watanabe, Zheng-Hua Tan, and Najim Dehak. Effectiveness of single-channel blstm enhancement for language identification. In *Interspeech 2018*, volume 2018-September, pages 1823–1827. ISCA, 9 2018. doi: 10.21437/Interspeech.2018-2458. [Cited on page 24]

[65] S. Spors. *Active Listening Room Compensation for Spatial Sound Reproduction Systems*. PhD thesis, Friedrich-Alexander-Universität Erlangen-Nürnberg, Jan 2006. [Cited on page 4]

[66] M.A. Stone and B. C. J Moore. Tolerable hearing aid delays. i. estimation of limits imposed by the auditory path alone using simulated hearing losses. *Ear and Hearing*, 20(3):182–192, 1999. [Cited on pages 3 and 25]

[67] Cees Taal, Richard C. Hendriks, R Heusdens, and Jesper Jensen. A short-time objective intelligibility measure for time-frequency weighted noisy speech. pages 4214 – 4217, 04 2010. doi: 10.1109/ICASSP.2010.5495701. [Cited on page 8]

[68] K. Tan, X. Zhang, and D. Wang. Real-time speech enhancement using an efficient convolutional recurrent network for dual-microphone mobile phones in close-talk scenarios. In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5751–5755, May 2019. doi: 10.1109/ICASSP.2019.8683385. [Cited on page 26]

[69] Alaa Tharwat. Classification assessment methods. *Applied Computing and Informatics*, 2018. ISSN 2210-8327. doi: https://doi.org/10.1016/j.aci.2018.08.003. [Cited on pages 33 and 34]

[70] M. Tu, V. Berisha, Y. Cao, and J. Seo. Reducing the model order of deep neural networks using information theory. In *2016 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, pages 93–98, July 2016. doi: 10.1109/ISVLSI.2016.117. [Cited on page 40]

[71] M. Valenti, S. Squartini, A. Diment, G. Parascandolo, and T. Virtanen. A convolutional neural network approach for acoustic scene classification. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 1547–1554, May 2017. doi: 10.1109/IJCNN.2017.7966035. [Cited on page 22]

[72] Steven Van Kuyk, W Kleijn, and Richard Christian Hendriks. An evaluation of intrusive instrumental intelligibility metrics. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 07 2018. doi: 10.1109/TASLP.2018.2856374. [Cited on page 8]

[73] Peter Vary and Rainer Martin. *Digital Speech Transmission: Enhancement, Coding And Error Concealment*. John Wiley & Sons, Inc., USA, 2006. ISBN 0470031743. [Cited on page 1]

[74] B. D. Van Veen and K. M. Buckley. Beamforming: a versatile approach to spatial filtering. *IEEE ASSP Magazine*, 5(2):4–24, April 1988. ISSN 0740-7467. doi: 10.1109/53.665. [Cited on pages 9 and 10]

[75] D. Wang and J. Chen. Supervised speech separation based on deep learning: An overview. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 26(10):1702–1726, Oct 2018. ISSN 2329-9290. doi: 10.1109/TASLP.2018.2842159. [Cited on pages 22 and 24]

[76] DeLiang Wang. *On Ideal Binary Mask As the Computational Goal of Auditory Scene Analysis*, pages 181–197. Springer US, Boston, MA, 2005. ISBN 978-0-387-22794-8. doi: 10.1007/0-387-22794-6_12. [Cited on page 2]

[77] DeLiang Wang and Guy J. Brown. *Computational Auditory Scene Analysis: Principles, Algorithms, and Applications*. Wiley-IEEE Press, 2006. ISBN 0471741094. [Cited on page 2]

[78] Zhong-Qiu Wang and DeLiang Wang. All-neural multi-channel speech enhancement. pages 3234–3238, 09 2018. doi: 10.21437/Interspeech.2018-1664. [Cited on page 12]

[79] Lonce Wyse. Audio spectrogram representations for processing with convolutional neural networks. *CoRR*, abs/1706.09559, 2017. [Cited on page 29]

[80] Matthew D. Zeiler. ADADELTA: An adaptive learning rate method. *CoRR*, abs/1212.5701, 12 2012. [Cited on page 27]

[81] Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 818–833, Cham, 2014. Springer International Publishing. ISBN 978-3-319-10590-1. [Cited on page 28]

[82] Xueliang Zhang, Zhong-Qiu Wang, and DeLiang Wang. A speech enhancement algorithm by iterating single- and multi-microphone processing and its application to robust asr. 03 2017. doi: 10.1109/ICASSP.2017.7952161. [Cited on page 12]

Appendix A

Acoustic Environment Simulation

In this chapter, the data generation process is described for simulating acoustic environments and generating datasets.

A.1 Signal Generation

In order to train a CNN and subsequently assess the performance of the own-voice retrieval system, a large amount of noisy speech signals is needed. These signals are generated by summing randomly selected waveforms from two data sets containing clean speech and noise. Following the observation model in (1.1), let the own-voice speech and noise sequences of the m'th channel be denoted by $s_m[n]$ and $w_m[n]$, respectively. To simulate the setup depicted in Figure 1.1, the sequences are convolved using two sets of impulse responses. The first set contains the *head-related impulse responses* (HRIR), each describing the acoustic path between a fixed point in space and one of the two microphones on a HA. These impulse responses are measured using a circular array of 16 loudspeakers, which are placed equidistantly spaced around a real person with a diameter of 3 meters at eye-height. The other set contains the *own-voice impulse responses* (OVIR), which describe the acoustic path between the HA-wearer's own voice and one of the microphones on a HA. Both sets were measured in a listening room. By letting h_m^{HRIR} denote the HRIR and h_m^{OVIR} denote the OVIR mentioned impulse responses, respectively, the noisy observations $y_m[n]$ are constructed as

$$\begin{aligned} y_m[n] &= s[n] * h_m^{\text{OVIR}}[n] + \sum_w [n] * h_m^{\text{HRIR}}[n] \\ &= x_m[n] + v_m[n], \end{aligned} \tag{A.1}$$

where $m \in \mathbb{N}$ denotes channel. By comparing the power of each sequence in (A.1), a gain $g \in \mathbb{R}$ can be introduced to appropriately scale one of the sequences to simulate

a desired SNR situation, that is

$$g(\mathrm{SNR}_{\mathrm{dB}}) = \sqrt{10^{\frac{-\mathrm{SNR}_{\mathrm{dB}}}{10}} \frac{\sigma_x}{\sigma_v}}, \tag{A.2}$$

where σ_x and σ_v denote the variances of $x_m[n]$ and $v_m[n]$, respectively. The signal generation is depicted in Figure A.1.

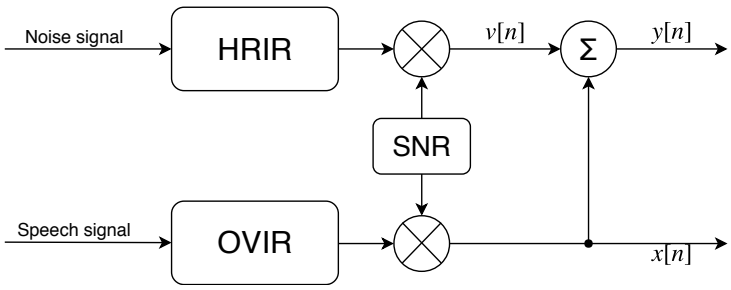


Figure A.1: Generation of noisy speech signals by scaling and summing the waveforms of the convolved noise and speech sequences.

Speech data

The speech data used for training and testing is extracted from the TIMIT speech corpus [61]. TIMIT contains 630 different readers of eight major dialects of American English, each reading ten phonetically rich sentences. The dataset is not gender-balanced, being 70% male readers and 30 % female. For training and validation, the predefined `TRAIN` subset will be used. For testing, the `DR1` dialect of the `TEST` will be used with a total duration of 259 seconds.

Noise data

Six different noise types are considered: on the bus (bus), cafeteria (caf), street junction (str) and pedestrian area (ped) from the CHiME3 dataset [2], together with speech shaped noise (ssn) from [36] and babble (bbl) noise generated by the author. The total duration of the signals is 270 minutes, from which 12 minutes are kept separate for testing purposes only. The noise types was chosen to cover a range of acoustic environments a HA wearer might be situated in.

A.2 Simulating Acoustic Environments

In [49] it was shown that the performance of the MVDR beamformer strongly depends on the incident angle of the desired source. For diffuse noise, it achieves the

optimal SNR gain in the endfire direction, whereas for point-source noise the performance depends on the angular separation between the point noise and desired source. This suggests that the simulated data should reflect different real-world scenarios with many different direction of arrivals.

The simulation of a noise point source is straight-forward from (A.1), where h_m^{HRIR} represents the impulse response from a single point in space to the m'th microphone. However, real-world acoustic environments rarely consist of just a single point source, but rather the cumulative effect of multiple sources impinging from different directions. In an attempt to accurately simulate this, $v_m[n]$ is modelled as a sum of multiple noise signals convolved with differently located HRIRs in space. Specifically, we model the six noise types in one of three ways. Speech-shaped noise (ssn) is modelled as a diffuse sound field, where a number of HRIRs equidistantly spaced on the circular speaker array are convolved with different realizations of the speech-shaped noise. Babble (bbl) noise is simulated as the sum of utterances from ten unique readers the `TRAIN` subset, each being convolved with a randomly chosen HRIR. Finally, the noise data from the CHiME3 dataset is modelled as a wavefront impinging the microphone array from a randomly chosen direction. Starting as the contribution of a single HRIR, the wavefront is simulated by including the contribution from nearby HRIRs as the wavefront passes the points from which the HRIRs were measured from. The contributions are synced in time by computing the delay in samples between the HRIRs, assuming the speed of sound is 314 m/s. When the wavefront reaches the microphone array, the remaining HRIRs are skipped. The concept is illustrated in Fig. A.2

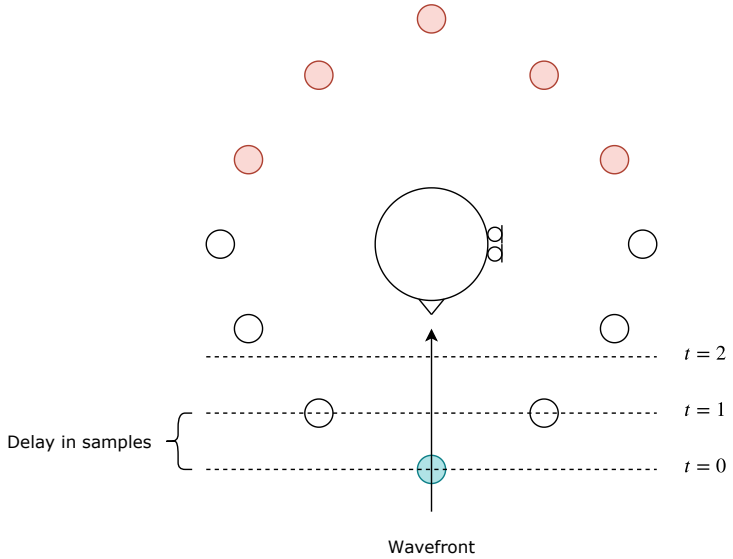


Figure A.2: A wavefront impinges from a randomly chosen direction. At $t = 0$, $v_m[n]$ consists of a single HRIR (marked with green) convolved with a single noise signal. At $t = 1$, the simulated wavefront passes two additional HRIRs, hence $v_m[n]$ is computed as the sum of three HRIRs and noise signals, which have been delayed accordingly. HRIRs marked with red are not used.